Journal of Artificial Intelligence Research ? (????) ?–? Submitted 11/12; published ??/??

Modular Structures and Atomic Decomposition in Ontologies — DRAFT, June 2, 2020, DO NOT CITE —

Chiara Del Vescovo CHIARA.DELVESCOVO@BBC.CO.UK
British Broadcasting Corporation
Data Management Team, Platform Data
MediaCityUK, Salford, M50 2BH, UK
Matthew Horridge MATTHEW.HORRIDGE@STANFORD.EDU
Stanford Center for Biomedical Informatics Research
Stanford University
California, 94305, USA
Bijan Parsia BIJAN.PARSIA@MANCHESTER.AC.UK
Uli Sattler ULI.SATTLER@MANCHESTER.AC.UK
School of Computer Science
University of Manchester
Manchester M13 9PL, UK
Thomas Schneider THOMAS.SCHNEIDER@UNI-BREMEN.DE
Fachbereich 3 Mathematik/Informatik
Universität Bremen
Postfach 330 440, 28334 Bremen, Germany
Haoruo Zhao HAORUO.ZHAO@MANCHESTER.AC.UK
School of Computer Science
University of Manchester
Manchester M13 9PL, UK

Abstract

With the growth of ontologies used in diverse application areas, the need for module extraction and modularisation techniques has risen. The notion of the *modular structure* of an ontology, which comprises a suitable set of base modules together with their logical dependencies, has the potential to help users and developers in comprehending, sharing, and maintaining an ontology. We have developed a new modular structure, called atomic decomposition (AD), which is based on modules that provide strong logical properties, such as locality-based modules. In this article, we present the theoretical foundations of AD, review its logical and computational properties, discuss its suitability as a modular structure, and report on an experimental evaluation of AD. In addition, we discuss the concept of a modular structure in ontology engineering and provide a survey of existing decomposition approaches.

1. Introduction

In artificial intelligence, knowledge representation is a prominent research topic that employs logic-based formalisms to make machines able to memorise, access, and process knowledge. In the past decades, the term *ontology* began to be widely used to denote a computer-processable description of the knowledge about a domain of interest, its concepts, and their interrelations.

A major class of ontology languages is based on *description logics*, for short *DLs* (Baader, Calvanese, McGuinness, Nardi, & Patel-Schneider, 2007), which generally are decidable fragments of first-order logic. Hence, an ontology can be viewed as a finite set of first-order formulas, called *axioms*. In particular, DLs provide the logical basis for the popular web ontology language OWL (Horrocks, Patel-Schneider, & van Harmelen, 2003; Cuenca Grau, Horrocks, Motik, Parsia, Patel-Schneider, & Sattler, 2008), which is a World Wide Web Consortium (W3C) standard.[1]

Ontologies are used in manifold applications in areas such as knowledge representation and management, the semantic web, biology, medicine, linguistics, and economics (Poli, Healy, & Kameas, 2010), where they define unambiguous vocabularies, represent background knowledge, and facilitate knowledge sharing and integration. Particularly in medicine, large and comprehensive ontologies have been developed, prominent examples being SNOMED CT, the Systematized Nomenclature of Medicine—Clinical Terms (Spackman, Campbell, & Côté, 1997) containing $>$ 400,000 logical axioms, and NCIT, the National Cancer Institute's Thesaurus (Golbeck, Fragoso, Hartel, Hendler, Oberthaler, & Parsia, 2003) with $>$ 200,000 logical axioms. Both these ontologies cover a wide range of subjects, such as diseases, symptoms, operations, treatments, devices and drugs.

Large ontologies pose serious challenges not only to the best optimised reasoners, but to all constituents of the ontology development process, such as navigation, editing, comprehension, and debugging. In addition, large ontologies are simply too broad for many purposes: for example, a user might be interested only in one medical specialisation, say cardiology. It is therefore desirable to design specific applications, such as diagnostic decision support for cardiologists, in a way that they focus only on the relevant part of the ontology. For this reason, and following the design principle *separation of concerns* borrowed from modular programming, strong efforts have been made to develop, maintain, and release ontologies organised into a *modular structure*, that is, a decomposition of an ontology into coherent fragments that may, or may not, loosely interact in a well-defined, meaningful way.

Module extraction and modularisation for ontologies have received great attention in recent years, in both theory and practice (Stuckenschmidt, Parent, & Spaccapietra, 2009). *A-priori* approaches aim at imposing a modular structure on an ontology to foster its modular development (Bao, Voutsadakis, Slutzki, & Honavar, 2009; Serafini & Tamilin, 2009; Cuenca Grau, Parsia, & Sirin, 2009), while *a-posteriori* approaches provide techniques for module extraction and modularisation of existing monolithic ontologies (see, e.g., Cuenca Grau, Parsia, Sirin, & Kalyanpur, 2006; Konev, Lutz, Walther, & Wolter, 2008; Cuenca Grau, Horrocks, Kazakov, & Sattler, 2009; Konev, Lutz, Ponomaryov, & Wolter, 2010).[2] The scenario described above and many similar ones clearly require a-posteriori approaches, simply because large ontologies currently exist in a monolithic, i.e., non-modular form. The a-posteriori approaches usually solve one of two tasks: GetOne, i.e., the extraction of a single module, or GetAll, i.e., the extraction of all modules or, more economically, the computation of the *modular structure* of an ontology, which comprises a representative subset of all modules together with their logical interactions. This modular structure has the potential

1. `https://www.w3.org/TR/owl2-overview/`
2. Throughout this article, we use the term "module" in a strict sense, assuming that a module provides logical guarantees such as coverage, i.e., the preservation of entailments over a given signature (Ghilardi, Lutz, & Wolter, 2006; Konev, Lutz, Walther, & Wolter, 2009).

to help users better understand the ontology, to aid ontology engineers in collaboratively developing it, and to optimise tool support, such as (incremental) reasoning. In this article, we are concerned with a-posteriori approaches of the GetAll category, i.e, the computation of the modular structure of a monolithic ontology.

The search for a semantically grounded modular structure can be traced back to the 1980s, when the fundamental concepts of *update* and *belief revision* of a logical theory were investigated (Alchóurron, Gärdenfors, & Makinson, 1985). In the classical belief revision task, the input is a theory, given by a finite set of *axioms* (formulas in propositional logic), and an *update*: a single axiom that is possibly inconsistent with the theory. The goal is to modify the theory such that the inconsistency is resolved, and to add the update. In this context, the following natural question arises: which part of the theory is affected by the update? Parikh (1999) defined the notion of a *signature decomposition* for a logical theory, which partitions the signature (i.e., the non-logical vocabulary) of the theory and thereby induces a partition of the theory into signature-disjoint parts. The last condition guarantees that parts which do not share terms with the update can remain unchanged. Intuitively, the parts can be regarded as distinct, independent "topics" of the theory.

Signature decomposition was refined by Konev et al. (2010) for OWL ontologies, triggered by the observation that certain terms occur in many axioms of an ontology, thus prohibiting a decomposition into signature-disjoint parts. For example, the role (binary relation) `hasPart` is typically used to define concepts belonging to various topics, such as anatomy, substances, vehicles, events. Konev et al.'s approach allows to treat terms like `hasPart` as logical symbols by manually grouping them into a separate signature Δ, and to then decompose the remainder of the theory's signature, coining the name *signature* Δ*-decomposition* for this technique.

Both approaches decompose the signature and not the ontology itself. They induce a decomposition of the ontology's axioms, but that decomposition is of limited value as a modular structure: (1) due to the signature disjointness, there is no (Parikh) or very little (Konev et al.) logical connection between the parts; hence the strict requirement of signature disjointness leads to rather shallow modular structures; (2) the parts are not necessarily subsets of the input ontology, i.e., rewriting of axioms is allowed and even required; (3) algorithms for computing the signature decomposition and its ontology decomposition are restricted to rather inexpressive logics (i.e., *DL-Lite* and $\mathcal{EL}$) but can still lead to an exponential number of parts compared to the size of the input ontology (Konev et al., 2010).

An approach to decomposing the ontology itself was introduced by Cuenca Grau et al. (2006); it consists in computing the finest possible $\mathcal{E}$*-connection* (Kutz, Lutz, Wolter, & Zakharyaschev, 2004) that is equivalent to the input ontology $\mathcal{O}$ (under certain assumptions). The result is a partition of the axioms of $\mathcal{O}$ with links between them, and an assignment of the terms in $\mathcal{O}$ to the parts of the partition. These parts are not necessarily signature-disjoint, but they have models of a certain shape: the extensions of any two terms assigned to distinct parts are disjoint, which nicely captures the intuition of distinct "topics" and the logical interactions between those, in the sense that the set of terms assigned to one part constitutes the topic of that part and the links indicate that the knowledge in $\mathcal{O}$ about one topic depends on knowledge in $\mathcal{O}$ about other topics. While this decomposition approach does not suffer from problems (2) and (3) described above, it still does not always yield a useful modular structure because certain modelling patterns allow only coarse-grained, "topic-unaware" decompositions.

In the search for a modular structure that does not suffer from the previously encountered problems, we (Parsia & Schneider, 2010) started to investigate *locality-based modules*, for short *LBMs* (Cuenca Grau, Horrocks, Kazakov, & Sattler, 2008) as a possible basis for a fine-grained, well-connected, and easily computable modular structure. We chose LBMs because they provide strong logical guarantees and can be computed efficiently. The most important logical guarantee is *coverage*: given an ontology and a user-selected set of terms Σ, called a *seed signature*, an LBM for Σ preserves all the ontology's entailments formulated in Σ. The seed signature can thus be seen as determining a specific "topic", and the corresponding LBM "covers" the ontology for this topic (but, in general, not for any different topic). LBMs share coverage with other module notions, but unlike those, the syntactic variants of LBMs can be computed efficiently in logics up to the DL underlying OWL.

The extraction of an LBM given Σ is a typical instance of the task GetOne. In order to use LBMs in GetAll, the most naïve approach would be to extract modules for all possible seed signatures. This is clearly unfeasible because the number of seed signatures is exponential in the size of the input ontology, and so is the number of modules: even though modules for distinct seed signatures often coincide, the exponential behaviour cannot be avoided in existing ontologies (Parsia & Schneider, 2010). It is hard to see how a 'modular structure' of exponential size could help with any task that is complicated for medium-to-large ontologies.

In order to obtain a useful modular structure despite these observations, we can distinguish *fake* modules—i.e., those that are simply unions of other modules—from *genuine* ones. As we have shown (Del Vescovo, Parsia, Sattler, & Schneider, 2011), there are only linearly many genuine modules, and they can be computed efficiently. While genuine modules can still overlap, they induce a partition of the input ontology together with a dependency relation between the parts, in a natural way. The resulting decomposition, as well as the overall technique, is called *atomic decomposition (AD)*; in previous work we laid its theoretical foundations (Del Vescovo et al., 2011) and discussed some aspects of its usefulness in practice (Del Vescovo, Gessler, Klinov, Parsia, Sattler, Schneider, & Winget, 2011).

This article gives a detailed account of AD and compares it with the preceding approaches to obtaining a modular structure. Its main contributions consist of

(§3) a qualitative analysis of the modular structures obtained by the previously described approaches, preceded by a discussion of the concept of a modular structure in ontology engineering and of the properties that a "good" modular structure should have;

(§4) a detailed presentation of the theoretical foundations of AD, extending our previous work (Del Vescovo et al., 2011) with a discussion of the modular structure induced by AD and with comments on using module notions other than LBMs as the basis of AD; b

(§5) an experimental evaluation of AD, updating our previous empirical study (Del Vescovo et al., 2011) with a representative up-to-date ontology corpus, and a corrected implementation.

In addition, we define the basic concepts around modules in §2 and conclude in §6.

This work is an extension to earlier work on atomic decomposition and modular structures of ontologies by some of the same authors: decompositions using $\mathcal{E}$-connections are described and investigated in (Cuenca Grau et al., 2009; Cuenca Grau, Parsia, & Sirin, 2006); in

(Del Vescovo et al., 2011), the theoretical foundations of ADs are laid; in (Del Vescovo et al., 2011), an early analysis of the structure of ADs of existing ontologies is described; (Klinov, Del Vescovo, & Schneider, 2012) discusses an approach to persist the AD; De Vescovo's dissertation (Del Vescovo, 2013) discusses both the theoretical foundations and early experimental evaluation of ADs, but both have been significantly enhanced in this paper. In particular, Section 2 and 5 are new, and Sections 3 and 4 are completely re-worked.

2. Preliminaries

We assume the reader to be familiar with description logic, in particular to have a good understanding of the description logics $\mathcal{ALC}$ or $\mathcal{SROIQ}$ (Baader, Horrocks, Lutz, & Sattler, 2017; Horrocks, Kutz, & Sattler, 2006) including their syntax and semantics, ontologies, interpretations and models, and entailment relation. In this section, we only fix the core notations of description logics and repeat some well-established key notions around module extraction (Ghilardi et al., 2006; Konev et al., 2009; Kontchakov, Pulina, Sattler, Schneider, Selmer, Wolter, & Zakharyaschev, 2009; Cuenca Grau et al., 2009).

2.1 Conservative extensions, inseparability, and modules

We start with the basic definitions underlying most logic-based notions of modules. Let $\mathcal{O}$ denote an ontology, i.e., a finite set of *axioms*. An axiom can be a general concept inclusion, a role inclusion, a concept assertion or a role assertion. We use $\mathsf{N_C}$ for a set of concept names and $\mathsf{N_R}$ a set of role names. A *signature* is a set $\Sigma \subseteq \mathsf{N_C} \cup \mathsf{N_R}$ of *terms*. Given a concept, role, axiom, or ontology X, the set of all terms occurring in X is called the *signature of* X, and denoted by $\widetilde{X}$.

We use SOL for second-order logic, and use $\mathcal{L}$ for a description logic or a subset of SOL. For an interpretation $\mathcal{I}$ and an ontology $\mathcal{O}$, we write $\mathcal{I} \models \mathcal{O}$ if $\mathcal{I}$ is a model of $\mathcal{O}$, and we use $\mathcal{I}|_\Sigma$ to denote the restriction of $\mathcal{I}$ to the signature Σ. The consequence relation of SOL and standard DLs is known to be *monotonic*, i.e., extending an ontology with additional axioms preserves its previous consequences.

Definition 2.1 (Ghilardi et al., 2006; Konev et al., 2009)**.** Let $\mathcal{O}$ be an $\mathcal{L}$-ontology, $\mathcal{M} \subseteq \mathcal{O}$, and Σ a signature. We say that

1. $\mathcal{O}$ is a *deductive* Σ*-conservative extension* (Σ*-dCE*) of $\mathcal{M}$ w.r.t. $\mathcal{L}$ if, for all $\mathcal{L}$-axioms α with $\widetilde{\alpha} \subseteq \Sigma$, it holds that $\mathcal{M} \models \alpha$ if and only if $\mathcal{O} \models \alpha$;
2. $\mathcal{O}$ is a *model* Σ*-conservative extension* (Σ*-mCE*) of $\mathcal{M}$ if $\{\mathcal{I}|_\Sigma \mid \mathcal{I} \models \mathcal{M}\} = \{\mathcal{J}|_\Sigma \mid \mathcal{J} \models \mathcal{O}\}$;
3. $\mathcal{M}$ is a *mCE-based (dCE-based) module for* Σ of $\mathcal{O}$ (w.r.t. $\mathcal{L}$) if $\mathcal{O}$ is a Σ-mCE (Σ-dCE) of $\mathcal{M}$ (w.r.t. $\mathcal{L}$);
4. If $\mathcal{M}$ is a dCE-based module for Σ w.r.t. $\mathcal{L}$, we also say that $\mathcal{M}$ *covers* or *provides coverage to* $\mathcal{O}$ for Σ.

Since $\mathcal{M} \subseteq \mathcal{O}$, the monotonicity of $\mathcal{L}$ implies that the "only if" direction of Point 1 in Definition 2.1 holds trivially. Furthermore, for a description logic or subset of SOL $\mathcal{L}$, every

mCE-based module is also a dCE-based module w.r.t. $\mathcal{L}$; the converse is not necessarily the case unless $\mathcal{L}$ is equally expressive as SOL.

In order to abstract from the requirement that $\mathcal{M}$ is a subset of $\mathcal{O}$, the notion of inseparability relation has been introduced.

Definition 2.2 (Konev et al., 2009)**.** Let $\mathcal{L}$ be a logic, $\mathcal{O}_1$ and $\mathcal{O}_2$ two $\mathcal{L}$-ontologies, and Σ a signature. We say that

1. $\mathcal{O}_1$ and $\mathcal{O}_2$ are *model inseparable w.r.t.* Σ, written $\mathcal{O}_1 \equiv_\Sigma^{\text{mCE}} \mathcal{O}_2$, if $\{\mathcal{I}|_\Sigma \,|\, \mathcal{I} \models \mathcal{O}_1\} = \{\mathcal{I}|_\Sigma \,|\, \mathcal{I} \models \mathcal{O}_2\}$;
2. $\mathcal{O}_1$ and $\mathcal{O}_2$ are *deductively inseparable w.r.t.* Σ *in* $\mathcal{L}$, written $\mathcal{O}_1 \equiv_\Sigma^{\text{dCE-}\mathcal{L}} \mathcal{O}_2$, if, for all $\mathcal{L}$-axioms η over Σ, we have that $\mathcal{O}_1 \models \eta$ if and only if $\mathcal{O}_2 \models \eta$.

The equivalence relations $\equiv_\Sigma^R$ are defined upon the notion $R \in \{\text{mCE}, \text{dCE-}\mathcal{L}\}$ which is called an *inseparability relation*. Whenever $\mathcal{L}$ is clear from the context, we will omit it.

Please note that other notions of inseparability relations can be defined (Konev et al., 2009; Sattler, Schneider, & Zakharyaschev, 2009; Kontchakov, Wolter, & Zakharyaschev, 2010), even though for the purposes of this paper we will consider only inseparability relations $R \in \{\text{mCE}, \text{dCE-}\mathcal{L}\}$.

The following property of *monotonicity* is of crucial importance for the inseparability relations just defined, and will have a deep impact on the properties of modules of interest in this paper.

Definition 2.3 (Kontchakov et al., 2009)**.** An inseparability relation R is called *monotonic* if, for all ontologies $\mathcal{O}_1, \mathcal{O}_2$ and $\mathcal{O}_3$ and each signature Σ, it satisfies the following conditions.

(M_{sig}) if $\mathcal{O}_1 \equiv_\Sigma^R \mathcal{O}_2$, then $\mathcal{O}_1 \equiv_{\Sigma'}^R \mathcal{O}_2$ for each $\Sigma' \subseteq \Sigma$;

(M_{axs}) if $\mathcal{O}_1 \subseteq \mathcal{O}_2 \subseteq \mathcal{O}_3$ and $\mathcal{O}_1 \equiv_\Sigma^R \mathcal{O}_3$, then $\mathcal{O}_1 \equiv_\Sigma^R \mathcal{O}_2$.

The inseparability relation $R = \text{mCE}$ is monotonic, which is an easy consequence of the monotonicity of SOL. Analogously, for every monotonic DL $\mathcal{L}$, $R = \text{dCE-}\mathcal{L}$ is monotonic.

Inseparability relations induce modules, while abstracting away from the above restriction to mCE and dCE, from a concrete language $\mathcal{L}$, and from the requirement $\mathcal{M} \subseteq \mathcal{O}$:

Definition 2.4. Let R be an inseparability relation, $\mathcal{M}$ and $\mathcal{O}$ two ontologies, and Σ a signature. We call $\mathcal{M}$

1. an *R_Σ-module of $\mathcal{O}$* if $\mathcal{M} \equiv_\Sigma^R \mathcal{O}$;
2. a *self-contained R_Σ-module of $\mathcal{O}$* if $\mathcal{M} \equiv_{\Sigma \cup \widetilde{\mathcal{M}}}^R \mathcal{O}$;
3. a *depleting R_Σ-module of $\mathcal{O}$* if $\mathcal{O} \setminus \mathcal{M} \equiv_{\Sigma \cup \widetilde{\mathcal{M}}}^R \emptyset$.

$\mathcal{M}$ is called a *minimal* (self-contained, depleting) R_Σ-module of $\mathcal{O}$ if $\mathcal{M}$, but no proper subset of $\mathcal{M}$, is a (self-contained, depleting) R_Σ-module of $\mathcal{O}$.

Definition 2.4 is a generalisation of the original definition by Kontchakov et al. (2009), which additionally assumed $\mathcal{M} \subseteq \mathcal{O}$. We have dropped this assumption because some of the decomposition approaches introduced later will induce modules that are, in general, not subsets of the input ontologies. However, the notion of a *minimal* module will be used in this article only in connection with modules that are subsets.

To ease notation, we will omit the symbol Σ from the notion R of an inseparability relation whenever the specific signature Σ is irrelevant.

2.2 Computational issues: a summary

Unfortunately, deciding whether a fragment $\mathcal{M}$ of an ontology $\mathcal{O}$ is an R_Σ-module for $R \in \{\text{mCE}, \text{dCE-}\mathcal{L}\}$, and thus the computation of a module, is computationally hard except for very lightweight ontology languages: while mCEs can be decided in PTime for acyclic $\mathcal{ELI}$ ontologies (Konev et al., 2008), they are decidable and coNExp-hard for some fragments of *DL-Lite* (Kontchakov et al., 2010), and undecidable already for general $\mathcal{EL}$ ontologies (Lutz & Wolter, 2010) and acyclic $\mathcal{ALC}$ ontologies (Konev, Lutz, Walther, & Wolter, 2013). Restricting the signature Σ to contain only concept names may lower the complexity somewhat or regain decidability for $\mathcal{EL}$ and $\mathcal{ALC}$, see (Konev et al., 2013, Table 1).

The computational behaviour of dCEs is slightly better but still prohibitive: deciding dCEs is ExpTime-complete for $\mathcal{EL}$ (Lutz & Wolter, 2010), 2ExpTime-complete for $\mathcal{ALC}$ (Ghilardi et al., 2006) and $\mathcal{ALCQI}$ (Lutz, Walther, & Wolter, 2007), and undecidable already for $\mathcal{ALCQIO}$ (Lutz et al., 2007) and thus for $\mathcal{SROIQ}$.

In face of the high computational complexity of identifying modules in general, various lines of research have been pursued which either restrict the ontology language, or provide useful approximations; the next subsection provides an overview.

2.3 Existing module notions

Specialised methods have been developed for inexpressive DLs; the MEX approach (Konev et al., 2008) extracts, under certain conditions, minimal self-contained and depleting modules from acyclic $\mathcal{ELI}$ terminologies in polynomial time; the method described by Kontchakov *et al.* (2009) does the same for *DL-Lite* ontologies, whereas the AMEX approach (Gatens, Konev, & Wolter, 2014) describes a closely related method for acyclic $\mathcal{ALCQI}$ terminologies.

Approximation methods have been designed for DLs as expressive as $\mathcal{SROIQ}$: the family of *locality-based modules*, for short *LBMs* (Cuenca Grau et al., 2008) are self-contained, depleting modules which are possibly non-minimal, i.e., may contain axioms that are not relevant to preserve entailments over the given *seed signature* Σ or the module's signature. There are various flavours of locality-based modules: there are *syntactic*-$\bot$ and *semantic*-$\emptyset$ modules as well as *syntactic*-$\top$ and *semantic*-Δ ones, as well as their nested version like the *syntactic*-$\top\bot^*$ module and more. For various expressive description logics, syntactic locality-based modules can be extracted in polynomial time (Cuenca Grau et al., 2008).

In more recent work, the notion of (syntactic) locality has been extended in at least two directions by introducing interesting data structures which, for a given ontology, represent relevant dependencies between terms and underlie the process of extracting a module. The first such extension uses (hyper)graphs to describe dependencies between signatures of sets of axioms, i.e., seed signatures of modules. This approach and the resulting reachability-based

modules (RBMs) have been developed by Suntisrivaraporn (2008) and extended to $\mathcal{SRIQ}$ and $\mathcal{SROIQ}$ by Nortjé, Britz, and Meyer (2013). Similarly to LBMs, RBMs come in three flavours $\bot$, $\top$, and $\top\bot^*$. While, for ontologies in a certain normal form, $\bot$-RBMs coincide with $\bot$-LBMs, the other two variants are usually subsets of the corresponding LBM variant and are neither self-contained nor depleting; however, all variants are robust under a number of interesting operations such as vocabulary restrictions and extensions (Nortjé et al., 2013). Furthermore, a different hypergraph representation and the same normal form have been used by Martín-Recuerda and Walther (2014) to compute $\bot$-LBMs more efficiently.

The second extension of locality-based modules represents dependencies between terms in a datalog programme. This approach, developed by Armas Romero, Kaminski, Cuenca Grau, and Horrocks (2016) for DLs up to $\mathcal{SROIQ}$, provides a whole 'suite' of module notions covering a wide range of inseparability relations. Some of these notions have been shown to be depleting and self-contained, and to satisfy further useful properties, such as the preservation of justifications (Armas Romero et al., 2016).

In some sense orthogonal to locality and its extensions, $\mathcal{E}$-connections (Kutz et al., 2004) give rise to another notion of self-contained modules, which will be discussed in more detail in Section 3.3.3.

As a result, we have various notions of modules with different properties, and various algorithms to extract these: some run in polynomial time, some only work on input ontologies of a restricted form but ensure minimality, some work on $\mathcal{SROIQ}$ ontologies but do not guarantee minimality. Various studies have investigated how the choice of a module notion affects the relative size of modules in practice (Del Vescovo, Klinov, Parsia, Sattler, Schneider, & Tsarkov, 2013; Nortjé et al., 2013; Gatens et al., 2014). Algorithms for extracting AMEX modules, LBMs, RBMs, and DBMs have been implemented in various tools, such as AMEX, the OWL API, CEL, and PrisM; links are provided in the original articles.

All module notions discussed so far are based on the requirement that a module of an ontology $\mathcal{O}$ be a subset of $\mathcal{O}$. However, in this article we also discuss modules in a broader sense, i.e., ontologies obtained by rewriting (parts of) $\mathcal{O}$. The signature decomposition approaches discussed in Sections 3.3.1 and 3.3.2 implicitly yield such modules, whose existence is ensured by certain interpolation properties of the underlying logic (Konev et al., 2010). In particular, uniform interpolants (UIs) can be used to yield (non-subset) modules with very similar logical guarantees, including Σ-inseparability. UIs have been studied widely for DLs of various expressivity (e.g., by ten Cate, Conradie, Marx, & Venema, 2006; Wang, Wang, Topor, Pan, & Antoniou, 2009b; Lutz & Wolter, 2011; Lutz, Seylan, & Wolter, 2012; Nikitina & Rudolph, 2014; Koopmann & Schmidt, 2014, 2015) and are closely related to (in fact, they are the dual notion of) *forgetting* (Reiter & Lin, 1994; Lang, Liberatore, & Marquis, 2003; Eiter, Ianni, Schindlauer, Tompits, & Wang, 2006; Konev, Walther, & Wolter, 2009; Wang, Wang, Topor, Pan, & Antoniou, 2009a; Zhao & Schmidt, 2017, 2018b; Chen, Alghamdi, Schmidt, Walther, & Gao, 2019a).

The signature decomposition approaches discussed in Sections 3.3.1 and 3.3.2 are related to UIs, but we are not aware of any work that addresses the computation of modules from a signature decomposition directly. All other decomposition approaches discussed in this article are concerned with modules that are subsets of the input ontology. Furthermore, the precise relationship between module-specific properties and UIs/forgetting-based modules are not yet known, except for the insight that module extraction can be used as an optimisation

to forgetting (Chen et al., 2019a). For this reason, a further discussion of UIs is beyond the scope of this article.

2.4 General remarks

In the following sections, we will focus mainly on description logics, and mainly on logics *without* nominals: this is due to the fact that nominals are rather ill-behaved in general—for example, they lead to the loss of nice model properties such as the tree model property and the property that models are closed under disjoint union—which causes serious problems for module extraction.

As indicated above, we consider module notions where each module is a subset of the ontology, and module notions where this is not necessarily the case. Please note that the former preserves the integrity of axioms, which is important for some applications, e.g., where axioms are annotated with their provenance. This preservation comes, however, at a cost: it is well known that the TBox $\mathcal{O} = \{C_i \sqsubseteq D \mid 1 \leq i \leq n\}$ is equivalent to its singleton counterpart $\mathcal{O}' = \{\top \sqsubseteq \bigsqcap_{1 \leq i \leq n} \neg C_i \sqcup D\}$, yet $\mathcal{O}$ has 2^n subsets all of which are $\bot$-locality based modules, whereas $\mathcal{O}'$ only has two subsets, $\emptyset$ and itself, and thus at most two subset modules. Hence subset-based module notions are *syntax dependent*, i.e., equivalent ontologies may lead to different, non-equivalent modules.

3. Qualitative analysis of existing modular structures

In this section we discuss the concept of modular structure in ontology engineering, in particular the properties that a "good" modular structure of an ontology should have so that it supports various knowledge engineering tasks. We start with a general discussion of these points in analogy to software engineering. Then we focus on the modular structure of ontologies, where some interesting problems arise from the fact that entailments behave like side effects that are difficult to control. Finally, we discuss existing approaches to modular structures and their behaviour with respect to the above mentioned properties.

3.1 Core concepts in modular engineering

Before looking at modular design in knowledge engineering, we discuss modularity in software design to clarify relevant concepts. In Software Engineering, the decomposition of pieces of code into suitable modules has been studied since the late 1960s. In 1972, Parnas recommends the use of *functional modules* (Parnas, 1972) , i.e., modules incorporating a function, rather than *sequential modules*, i.e., modules determined by the temporal sequence of actions to perform. These modules are combined via interacting with the modules' *interfaces*, that is, their input and output parameters. This allows for a *separation of concerns* where distinct modules implement distinct, independent functionalities. The system obtained is then independent of the specific implementation "inside" each of its modules; in other words, a module has a functional core which is (relatively) independent of other parts of the code. Nowadays this property is called *encapsulation*. A closely related concept is that of *information hiding*: not only is the system agnostic as to how a given module implements its functionality, it also cannot affect this implementation. That is, the module's functionality is protected from changes outside the module, and the rest of the system knows of the

module's interface only. Furthermore, a good module should implement one functionality rather than several independent ones; thus a good module has a high *cohesion*, with all its parts being strongly related. A system with a module of low cohesion can usually be improved by splitting this module into several ones. As a consequence, modules need to be able to contain (or import) others, which naturally leads to a *dependency structure* between modules: module A depends on module B if A imports B or requires B's presence via some other mechanism to function correctly.

The advantages of a good modular design are clear: a module is usually a small part of the whole (software) system, and thus easier to work with; different people can work on different modules independently and only need to agree on changes to interfaces or functionality; and modules can be reused, i.e., used in different software systems.

The main focus in software engineering was placed on the mechanisms to realise encapsulation and specify/work with interfaces, on methodologies to build systems in a suitable modular way where modules are of suitable size and structure, in particular of a hierarchical or nested structure, and on reusability of existing modules. There has been some work on automated support for code refactoring, in particular for refactoring "spaghetti code" (monolithic code) into suitable modules, and state-of-the-art IDEs provide this kind of tool support. This support, however, usually leaves existing classes intact and only moves these into other packages, i.e., it starts from an existing modular structure and rearranges it without introducing new modules or merging modules.

In logic-based knowledge engineering, a modular design has potentially the same benefits as in software engineering, so we are discussing the concepts developed for modular software systems in the context of ontology engineering. For modules of ontologies, various properties have been introduced in the literature (Cuenca Grau et al., 2009; Konev et al., 2009; Kontchakov et al., 2009), see Section 2.1. We now relate these to those developed for modular software systems.

First, throughout this paper, we focus on the problem of *refactoring* a given ontology into a suitable set of modules, i.e., on the problem of automatically turning a monolithic "spaghetti-code" ontology into one that exhibits a good modular structure. One of the problems here is that an ontology itself exhibits very little structure: instead of variables, functions, classes, methods, loops, or function calls, we only have a (unordered) set of axioms in which terms (co-)occur. In addition, there are many ways to rewrite axioms into equivalent ones and change the co-occurrence relation between terms, e.g., $A \sqsubseteq (B \sqcap C)$ is equivalent to $A \sqsubseteq B,\ \ A \sqsubseteq C$. As we have seen in Section 2.2, the extraction of a single minimal module for a given signature is a complex problem which may not even be computable, depending on the exact properties of the desired module and the underlying logic. The task of computing a suitable *range* of modules for a given ontology is even more difficult, but will be at the heart of the envisaged refactoring support and thus its complexity needs to be considered.

Second, let us discuss the notion of a module's *interface*: from the above understanding of an interface as a set of parameters that ensures encapsulation, an obvious candidate for an interface of an ontology module is the set of terms Σ such that $\mathcal{M}$ is R_Σ-inseparable from $\mathcal{O}$. In this case, $\mathcal{M}$ encapsulates the full functionality of $\mathcal{O}$ regarding terms from Σ. There are, however, two problems with this idea. Firstly, this interface would not be unique.

Example 3.1. Consider the following ontology $\mathcal{O}$:

$$\alpha : A \sqsubseteq B \sqcap (C \sqcup D) \qquad \beta : C \sqsubseteq D \qquad \gamma : B \sqsubseteq D$$

Then, $\mathcal{M} = \{\alpha, \beta\}$ preserves all entailments over both $\Sigma_1 = \{A, B\}$ and $\Sigma_2 = \{A, C, D\}$, and so it would make sense to consider Σ_1 and/or Σ_2 as *interfaces* for $\mathcal{M}$. However, $\mathcal{M}$ does not preserve all entailments over $\Sigma_1 \cup \Sigma_2$; e.g., the entailment $B \sqsubseteq D$ is lost.

Secondly, we would need to record, or reconstruct, for each module, the sets of terms it covers: this would either lead to a more complex structure (of sets of axioms with their interfaces) or be computationally costly. As an alternative, we can require modules to be *self-contained*, i.e., to cover all the terms they mention. In Example 3.1, the smallest self-contained subset of $\mathcal{O}$ that preserves all entailments over Σ_1 is $\mathcal{O}$ itself.

The notion of modules with explicit interfaces has been discussed in the literature (Bao et al., 2009; Cuenca Grau et al., 2009; Serafini & Tamilin, 2009); here, we will concentrate on self-contained modules since, to the best of our knowledge, there is no logically sound approach to decomposing or refactoring ontologies that yields explicit interfaces.

Coverage guarantees that, for a given signature, we can use its module instead of the whole ontology because the module preserves all entailments over this signature. That is, coverage is closely related to *encapsulation*: a covering module does not rely on external information to fulfil its functionality.

Modules being *depleting* is also closely related to *encapsulation*: modules providing (only) coverage are called *weak modules* by Konev et al. (2008) because coverage alone does not guarantee encapsulation: it can happen that the same logical consequences of a module $\mathcal{M}$ are entailed also by $\mathcal{O} \setminus \mathcal{M}$, as described in the following example.

Example 3.2. Consider the following ontology.

$$\begin{aligned}
\mathsf{Sea} = \{ \quad & \exists \mathtt{hasFin}.\top \sqsubseteq \mathtt{Fish}, \\
& \mathtt{Dolphin} \sqsubseteq \exists \mathtt{hasCaudalFin}.\top, \\
& \mathtt{hasCaudalFin} \sqsubseteq \mathtt{hasFin} \qquad \} \cup \mathsf{Dolphin}, \text{ where} \\
\\
\mathsf{Dolphin} = \{ \quad & \mathtt{Dolphin} \sqsubseteq \mathtt{Fish}, \\
& \mathtt{Mammal} \sqsubseteq \neg \mathtt{Fish}, \\
& \mathtt{Dolphin} \sqsubseteq \mathtt{Mammal} \qquad \}
\end{aligned}$$

Clearly, Dolphin is a logical module of Sea for the signature $\Sigma = \{\mathtt{Dolphin}\}$, and thus preserves the entailment $\eta : \mathtt{Dolphin} \sqsubseteq \bot$. Now consider that we want to repair this unsatisfiability: we remove the axiom $\alpha : \mathtt{Dolphin} \sqsubseteq \mathtt{Fish}$ from Dolphin to yield Dolphin′ and Dolphin′ no longer entails η. However, replacing Dolphin with Dolphin′ in Sea results in an ontology that still entails η, i.e., our "modular" repair did not work globally. This is due to the fact that Dolphin is not a depleting module and thus does not *encapsulate* all our knowledge about Dolphins.

The definition of a depleting module would need to be adapted for cases where a module $\mathcal{M}$ is not a subset of $\mathcal{O}$: we would need to define an alternative to the extraction of a module, i.e., to $\mathcal{O} \setminus \mathcal{M}$, via a suitable rewriting mechanism.

Self-containment, which has already been mentioned above, guarantees that all terms in a module are equal, regardless of whether they are in the seed signature of this module or not. Given the above observations regarding a module's interface, a module needs to be self-contained to ensure that *all* its terms can reasonably be used as its interface. In other words, if a module is *not* self-contained, then it is only *functional* with respect to its seed signature, but possibly not for other terms.

Finally, it is not hard to see that modules—in particular when they are subsets of an ontology—can contain other modules, and thus may form a nested structure. One module containing another one can be seen as the former depending on the latter, and thus this containment relation induces a natural *dependency relation* between modules. Alternatively, in case where modules are subsets of an ontology, we can consider the partition induced by such nested modules. This partition consists of a set of "atoms" that we can combine to form our original set of modules. In OWL, we can store each atom as an OWL document and combine our modules via suitable "imports" statements. There is a direct one-to-one relation between this imports relation and the above mentioned dependency relation.

Next, we discuss two notions from software engineering that are more difficult to transfer to ontology modules. *Cohesion* is a notion that depends on what we mean by a module's parts and their relatedness. For ontology modules, we could say that a module is of low cohesion if it contains axioms that are not strongly related. While it is difficult to define precisely what strong (logical) relatedness means, it is possible to illustrate some extreme cases. For example, the weakest relatedness possible occurs between modules sharing neither axioms nor terms. But even if two axiom-disjoint modules share signature to some extent, they may still be widely unrelated, e.g., when the shared terms are only top-level concepts. Even two modules sharing axioms may be widely unrelated, e.g., when they share only tautologies. Consequently, it is reasonable to consider the union of two axiom- and term-disjoint modules to be of very low cohesion, and the union of two substantially different (for example, axiom- but not term-disjoint) modules to be of low cohesion.

Regarding *information hiding*, we first note that, if modules are mere sets of axioms, we have no way to specify which terms are "private" to a module; see above the discussion of interfaces. Hence in this setting, there can be no true support for information hiding. We can still ask, however, whether terms from a module are used *safely* in the rest of the ontology, i.e., whether they are only used (but not changed) or whether their meaning is changed in axioms outside the module.

The following example shows how information hiding or safe usage of terms can be tricky.

Example 3.3. Let us consider the following ontology.

$$\mathsf{Dolphin}' = \{\ \mathtt{Dolphin} \sqsubseteq (\exists \mathtt{hasPart}.\mathtt{CaudalFin}),\quad \mathtt{Fish} \sqsubseteq \mathtt{Vertebrate}\ \}$$

Now suppose we now want to add $\mathsf{Dolphin}'$ to our own ontology Fish:

$$\mathsf{Fish} = \{\ \mathtt{Fin} \sqsubseteq (\forall \mathtt{hasPart}^{-}.\mathtt{Fish}),\quad \mathtt{CaudalFin} \sqsubseteq \mathtt{Fin}\ \}$$

Looking only at the signature and observing that our Fish ontology does not relate terms from $\mathsf{Dolphin}'$ with each other, it might seem that the Fish ontology uses the terms from

$\mathsf{Dolphin}'$ in a safe way. However, taking the logical interactions into account, it turns out that $\mathsf{Dolphin}' \cup \mathsf{Fish} \models \texttt{Dolphin} \sqsubseteq \texttt{Fish}$, i.e., adding $\mathsf{Dolphin}'$ to Fish causes new entailments over the signature of $\mathsf{Dolphin}'$, and thus Fish does *not* use the terms from $\mathsf{Dolphin}'$ safely.

Fortunately, inseparability can also be used to formulate safety: given an inseparability relation R, we can say that $\mathcal{O}$ can *safely use* $\mathcal{M}$ if $\mathcal{O} \equiv^R_{\widetilde{\mathcal{M}}} \emptyset$, i.e., if $\mathcal{O}$ does not know anything about the terms in $\mathcal{M}$ (Jiménez-Ruiz, Cuenca Grau, Sattler, Schneider, & Berlanga Llavori, 2008).

3.2 Core Notions of modular ontology structures

Motivated by the discussion in the previous section, we will briefly summarise the core notions related to modular structures of ontologies.

In what follows, we use R for an inseparability relation and say that an *R-module* $\mathcal{M}$ of an ontology $\mathcal{O}$ is an ontology that is R-inseparable w.r.t. $\widetilde{\mathcal{M}}$ from $\mathcal{O}$. In other words, $\mathcal{M}$ is *self-contained*, but not necessarily a subset of $\mathcal{O}$, i.e., a module can be obtained by rewriting some axioms in $\mathcal{O}$: for example, one could split an axiom $A \sqsubseteq B_1 \sqcap B_2 \in \mathcal{O}$ into two axioms/modules $A \sqsubseteq B_i \in \mathcal{M}_i$. Also, depending on the notion of module, there does not have to be a seed signature Σ that determines $\mathcal{M}$. If there is a seed signature, then different seed signatures $\Sigma' \neq \Sigma$ may lead to the same module, which is then said to *cover* both these signatures, as well as $\widetilde{\mathcal{M}}$ given that $\mathcal{M}$ is self-contained.

As discussed in Section 2, various notions of modules fall into this category, including any form of semantic or syntactic locality-based modules, MEX modules, or nested modules like the $\top\bot^*$ module.

Let $\mathcal{O}$ be an ontology and x a module notion, which can be understood as a function that assigns an R_Σ-module $\mathcal{M}_\Sigma \subseteq \mathcal{O}$ to every signature Σ, for a fixed inseparability relation R. We call $\mathcal{M}_\Sigma$ the *x-Σ-module of $\mathcal{O}$*; it does not need to be the smallest R_Σ-module of $\mathcal{O}$. Ideally, there is an algorithm for computing $\mathcal{M}_\Sigma$ given Σ and $\mathcal{O}$ (see, e.g., Cuenca Grau et al., 2008; Konev et al., 2008).

A *modular structure* for x of $\mathcal{O}$ is a pair $(\mathfrak{G}, \rightarrow)$ where $\mathfrak{G}$ is a set of modules of $\mathcal{O}$ and $\rightarrow$ is a binary *dependency relation* between elements of $\mathfrak{G}$. A modular structure is also associated with a *module generation mechanism* that describes how to generate, for a given signature Σ the x-Σ-module $\mathcal{M}$ of $\mathcal{O}$ from $(\mathfrak{G}, \rightarrow)$.

The simplest form of the dependency relation is the subset relation - and this is also the one taken in all notions of modular structures discussed in this paper. In this case, and where modules are subsets of the ontology, $\mathfrak{G}$ gives rise to a *partition* of $\mathcal{O}$ into *atoms* where each module is a (disjoint) union of atoms. The module generation mechanism can be rather simple, for example, by taking the union of all those $\mathcal{M}_i \in \mathfrak{G}$ with $\widetilde{\mathcal{M}_i} \cap \Sigma \neq \emptyset$. It could also be a complex mechanism, with the extreme being the computation of the x-Σ-module from scratch.

We call the modules in $\mathfrak{G}$ *basic* modules, and those x-modules that are not basic *derived*.

As discussed earlier, the basic modules should be of high cohesion and are thus likely to be small; consequently modular structures should tend to have a high granularity. Given that the basic modules should form a reasonable basis for all x-modules, we expect $\mathfrak{G}$ to be minimal, i.e., to contain only those modules that cannot be derived via the module generation mechanism.

3.3 Existing approaches to determining the modular structure

In this subsection we describe three existing approaches to specifying the modular structure of an ontology that provide logical guarantees such as coverage. These approaches are Parikh's signature splitting (Parikh, 1999), signature Δ-decomposition (Konev et al., 2010), and decomposition based on $\mathcal{E}$-connections (Cuenca Grau et al., 2006). All these approaches consist in computing a partition of either the ontology (i.e., its set of logical axioms) or the ontology's signature (i.e., the set of all entities occurring in it). The last two of these approaches additionally include a non-trivial dependency relation between the modules. We will also discuss a fourth approach, namely an early attempt at identifying a modular structure based on *all* x-Σ-modules of an ontology (Parsia & Schneider, 2010) for a given module notion x, which has eventually led us to the notion of atomic decomposition as described in Section 4.

We describe each of these approaches and discuss some examples; then we discuss its relevant properties: the properties of the (basic and derived) modules, the cohesion of the basic modules, the dependency relation and thus granularity of the structure, the module generation mechanism, and the computational properties of identifying a modular structure. We also discuss how the different approaches relate to each other.

3.3.1 Parikh's Signature Splitting

The logical formalisation of this first notion of modular structure was introduced by Parikh (1999) in the context of belief revision. Its application scenario can be described generally as follows. One wants to revise an ontology[3] $\mathcal{O}$ by adding a new piece of knowledge that contradicts (some entailment of) $\mathcal{O}$. Parikh's splitting approach is intended to address the following question: to identify contradictions, do we have to check the new knowledge against all of $\mathcal{O}$, or can we make use of some kind of safety guarantee that allows us not to touch those parts that are irrelevant for this new finding? The approach is based on the simple observation that, under certain conditions, two logical formulas (or axioms of an ontology) that have disjoint signatures cannot logically interact with each other. Consequently, if $\mathcal{O}$ can be split into signature-disjoint parts, these parts do not logically interact with each other, and not all parts are relevant for freshly added knowledge. Parikh devises a principled way to perform such a splitting.

Definition 3.4 (Parikh, 1999)**.** Let $\mathcal{L}$ be a fragment of SOL, $\mathcal{O}$ an ontology in $\mathcal{L}$, and $\{\Sigma_1, \ldots, \Sigma_n\}$ a partition of $\widetilde{\mathcal{O}}$. We say that $\Sigma_1, \ldots, \Sigma_n$ *split* $\mathcal{O}$ in $\mathcal{L}$ if there are ontologies $\mathcal{O}_1, \ldots, \mathcal{O}_n$ in $\mathcal{L}$ with $\widetilde{\mathcal{O}_i} = \Sigma_i$, $i \leq n$, whose union is logically equivalent[4] to $\mathcal{O}$. We then call $\{\Sigma_1, \ldots, \Sigma_n\}$ an $\mathcal{O}$*-splitting* and $\mathcal{O}_1, \ldots, \mathcal{O}_n$ a *realization* of $\{\Sigma_1, \ldots, \Sigma_n\}$.

Clearly, every ontology $\mathcal{O}$ has at least one $\mathcal{O}$-splitting, $\{\widetilde{\mathcal{O}}\}$. In general and if no further assumptions are made, any $\mathcal{O}$ can have several splittings. However, in SOL there is always a unique *finest* $\mathcal{O}$-splitting: the component-wise intersection of two splittings is finer and has again a realization; the latter is not guaranteed in all fragments of SOL, see also the discussion in Section 3.3.2. More precisely, a splitting $\{\Sigma_1, \ldots, \Sigma_n\}$ *refines* a splitting $\{\Pi_1, \ldots, \Pi_m\}$ if

3. Parikh's original definition applies to logical theories in SOL. It should be clear that we can continue to use the notion of an ontology instead.
4. In the following, we use *equivalent* for *logically equivalent*. i.e., for having the same models.

both are distinct and every Π_i is the union of one or more Σ_j. For example, $\{\Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4\}$ refines $\{\Sigma_1 \cup \Sigma_2 \cup \Sigma_3, \Sigma_4\}$. A *finest* $\mathcal{O}$-splitting is an $\mathcal{O}$-splitting that refines every other $\mathcal{O}$-splitting.

Theorem 3.5 (Parikh, 1999)**.** *Every ontology $\mathcal{O}$ has a unique finest $\mathcal{O}$-splitting in SOL.*

Modules and their properties. In Parikh's splitting approach, a logical module of $\mathcal{O}$ is an ontology $\mathcal{O}_1$ such that the signature partition $\{\widetilde{\mathcal{O}}_1, \widetilde{\mathcal{O}} \setminus \widetilde{\mathcal{O}}_1\}$ splits $\mathcal{O}$. We call the $\mathcal{O}_i$ from Definition 3.4 the *Parikh modules* of $\mathcal{O}$ and their collection $\{\mathcal{O}_1, \ldots, \mathcal{O}_n\}$ the *Parikh modularisation* of $\mathcal{O}$. It has to be pointed out that the $\mathcal{O}_i$ are not required to be subsets of $\mathcal{O}$. Furthermore, they have to be considered modulo logical equivalence, which we do from here on. Since Parikh's original work is focusing on SOL, any ontology (finite set of formulae) can be equivalently expressed by a single formula. Hence, it may be the case that the set $\mathcal{O}$ is a singleton, but Parikh's splitting is non-trivial.

Example 3.6. Let us consider the singleton ontology:

$$\mathsf{One} = \{\top \sqsubseteq (\neg A_1 \sqcup B_1) \sqcap (\neg A_2 \sqcup B_2) \sqcap \cdots \sqcap (\neg A_n \sqcup B_n)\}$$

One can be rewritten into the logically equivalent ontology

$$\mathsf{All} = \{A_1 \sqsubseteq B_1,\ A_2 \sqsubseteq B_2,\ \ldots,\ A_n \sqsubseteq B_n\}$$

Hence, the signature partition $\Sigma_1 = \{A_1, B_1\},\ \ldots,\ \Sigma_n = \{A_n, B_n\}$ splits One.

It is easy to see that every Parikh module $\mathcal{O}_i$ of an ontology $\mathcal{O}$ provides *coverage* and is *self-contained*: this immediately follows from Definition 3.4: $\widetilde{\mathcal{O}_i}$ is disjoint from the signatures of the remaining $\mathcal{O}_j$, and the union of all $\mathcal{O}_j$ is logically equivalent to $\mathcal{O}$. In general, the $\mathcal{O}_i$ are *not depleting* because they are not necessarily subsets of $\mathcal{O}$; see the remark in Section 3.1.

To discuss *cohesion*, we use two simple example ontologies.

Example 3.7. Let us consider the following ontology.[5]

$$\begin{aligned}\mathsf{Tree} = \{ \quad \texttt{Tree} &\sqsubseteq \exists \texttt{isMadeFrom.Seed},\\ \texttt{Seed} &\sqsubseteq \exists \texttt{isMadeFrom.Fruit},\\ \texttt{Fruit} &\sqsubseteq \exists \texttt{isMadeFrom.Flower},\\ \texttt{Flower} &\sqsubseteq \exists \texttt{isMadeFrom.Branch},\\ \texttt{Branch} &\sqsubseteq \exists \texttt{isMadeFrom.Tree},\\ \texttt{Mountain} &\sqsubseteq \exists \texttt{isMadeFrom.Soil} \quad \}\end{aligned}$$

Parikh's splitting consists of a single part containing all the terms in Tree. However, there is a clear logical difference between the set of terms $\Sigma_1 = \{\texttt{Tree}, \texttt{Seed}, \texttt{Fruit}, \texttt{Flower}, \texttt{Branch}\}$ and the set $\Sigma_2 = \{\texttt{Mountain}, \texttt{Soil}\}$: by interpreting `Tree` as the empty set, all the other terms in Σ_1 are unsatisfiable, whilst there is no constraint in the interpretation of the terms in Σ_2. Similarly, by interpreting `Soil` as the empty set we get that `Mountain` is unsatisfiable, whilst there is no constraint in the interpretation of the terms in Σ_1. However, we cannot separate these two sets because they share the term `isMadeFrom`, so that Tree cannot be rewritten as the union of two signature-disjoint ontologies.

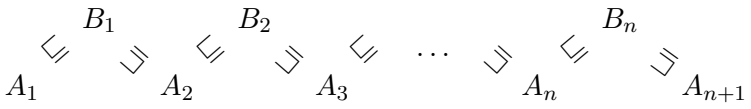
5. This ontology is an adaptation and translation into DL of some fragments from the Italian song *Ci vuole un fiore* (It requires a flower), by G. Rodari, L. Enriquez, and S. Endrigo.

Example 3.7 shows that the induced notion of cohesion is rather loose. Indeed, two terms A and B belong to the same module if there is an arbitrarily long sequence of intermediate terms $\{A_i \mid i \leq n, n \in \mathbb{N}\}$ such that A interacts (in the model-theoretic sense described above) with A_0, A_i interacts with A_{i+1} for all $i \in \{0, \dots n\}$, and A_n interacts with B. However, A and B may still not directly interact with each other. This situation is described in the following example.

Example 3.8. Let us consider the family Zigzag of ontologies, defined by

$$\mathsf{Zigzag}_n = \{A_1 \sqsubseteq B_1,\ A_2 \sqsubseteq B_1,\ A_2 \sqsubseteq B_2,\ \dots,\ A_n \sqsubseteq B_n,\ A_{n+1} \sqsubseteq B_n\},$$

with the following graphical representation.



For $n \geq 1$, the concepts A_1 and A_{n+1} are completely unrelated since Zigzag_n has only trivial entailments over the signature $\{A_1, A_{n+1}\}$. However, A_1 and A_{n+1} still belong to the same Parikh module.

Regarding information hiding, we note that, under certain circumstances,[6] distinct Parikh modules are completely unrelated, i.e., they share neither axioms nor terms. As a consequence, we can take the union of different modules and this will not affect the meaning of any terms described in any of them, and thus they can always import/be imported safely.

Modular structure and its properties. Thanks to Theorem 3.5, we can define a modular structure of an ontology $\mathcal{O}$ as follows: the basic modules are those $\mathcal{O}_i$ that determine the finest $\mathcal{O}$-splitting; the derived modules are their unions. The latter coincide with the Parikh modules; hence the basic modules are indeed representative of all Parikh modules. The module generation mechanism is simple: given an ontology $\mathcal{O}$ with the basic modules $\mathcal{O}_1, \dots, \mathcal{O}_n$ and a signature Σ, the Parikh-Σ-module of $\mathcal{O}$ is the union of all $\mathcal{O}_i$ with $\widetilde{\mathcal{O}_i} \cap \Sigma \neq \emptyset$.

There is no logical interaction between any two distinct basic modules because of their disjoint signatures. Hence the *dependency relation* is empty (i.e., the dependency graph consists of n isolated nodes). Furthermore, the *granularity* of the modular structure varies in the same way as the cohesion of its modules does, see the examples above.

Computational properties. Parikh's work (1999) does not study computational properties of signature decomposition or of computing a module. However, some insights can be transferred from the analysis of signature Δ-decompositions (Konev et al., 2010), which generalize Parikh splittings and are discussed in the following subsection.

3.3.2 Signature Δ-Decomposition

The modules induced by the finest $\mathcal{O}$-splitting in Parikh's approach have proved to be too coarse-grained for both Tree and Zigzag_n: both ontologies have only one module that contains all axioms although, intuitively, each ontology consists of several independent fragments.

6. This is true for propositional logic and all DLs where models are preserved under disjoint unions, i.e., for the $\mathcal{SHIQ}$ family or DLs without nominals.

Consequently, Parikh splittings may suffer from low cohesion. This problem can be overcome by imposing that any (part of an) ontology is coherent if its signature cannot be decomposed into disjoint subsets *by discarding some special, common terms*. These common terms can be distinctive for the whole domain underlying the knowledge represented in the ontology, but they can be used in different contexts, thereby representing substantially different logical relations. In Example 3.7, `isMadeFrom` is such a special term: it is obviously distinctive for the whole domain, but it is used in two different context—one related with plants and the other with landforms.

Ponomaryov (2008) has proposed and formalized this idea by introducing *signature* Δ-*decompositions*. The approach has been further investigated by Konev et al. (2010). Its main difference compared to Parikh's $\mathcal{O}$-splittings consists in considering an additional signature Δ that comprises the "special, common terms" and treating all these terms as if they were logical symbols. That is, the signature of an ontology $\mathcal{O}$ (originally: a finite set of SOL formulas) is decomposed into disjoint sub-signatures Σ_i, such that there exist ontologies $\mathcal{O}_i$ over the signatures $\Sigma_i \cup \Delta$ whose union is equivalent to $\mathcal{O}$.

Definition 3.9 (Konev et al., 2010)**.** Let $\mathcal{L}$ be a fragment of SOL, $\mathcal{O}$ an ontology in $\mathcal{L}$, and $\Delta \subseteq \widetilde{\mathcal{O}}$. A partition $\Sigma_1, \ldots, \Sigma_n$ of $\widetilde{\mathcal{O}} \setminus \Delta$ is called a *signature* Δ-*decomposition of* $\mathcal{O}$ *in* $\mathcal{L}$ if there are ontologies $\mathcal{O}_1, \ldots, \mathcal{O}_n$ in $\mathcal{L}$ such that

- $\widetilde{\mathcal{O}_i} \subseteq \Sigma_i \cup \Delta$ for $i = 1, \ldots, n$ and
- $\mathcal{O}_1 \cup \ldots \cup \mathcal{O}_n \equiv \mathcal{O}$.

$\mathcal{O}_1, \ldots, \mathcal{O}_n$ is called an $\mathcal{L}$-*realization* of $\Sigma_1, \ldots, \Sigma_n$ for $\mathcal{O}$.

In SOL a signature Δ-decomposition always exists and, as in Parikh's approach, is unique.

Theorem 3.10 (Konev et al., 2010)**.** *Every ontology* $\mathcal{O}$ *has a unique finest* Δ-*decomposition of* $\mathcal{O}$ *in SOL.*

As in Parikh's approach, the main reason for Theorem 3.10 being true is the fact that, given two distinct Δ-decompositions, one can always construct a finer partition by intersecting their components pairwise. In SOL that partition always has a realization and is thus a Δ-decomposition; this need not be the case in fragments of SOL.

Even though there is always a unique finest Δ-decomposition in SOL, the ontologies realizing it need not be uniquely determined, even modulo logical equivalence, since different realisations can spread $\mathcal{O}$'s knowledge about Δ in different ways across the $\mathcal{O}_i$s. To reflect the intuition that Δ is a set of "special, common terms", Konev et al. introduce realisations $\mathcal{O}_1, \ldots, \mathcal{O}_n$ for $\mathcal{O}$ where each $\mathcal{O}_i$ includes complete knowledge about the terms from Δ, i.e., $\mathcal{O}_i \equiv_\Delta \mathcal{O}_j$ and $\mathcal{O}_i \equiv_\Delta \mathcal{O}$ in $\mathcal{L}$ for all $i, j \leq n$. We call these realizations Δ-*encapsulating*. Konev et al. prove that being Δ-encapsulating is a sufficient condition for the uniqueness of realisations for many fragments $\mathcal{L}$ of SOL, including several DLs.

Definition 3.11 (Konev et al., 2010)**.** A fragment $\mathcal{L}$ of SOL *has unique decomposition realizations (UDR)* if, for all consistent $\mathcal{L}$-ontologies $\mathcal{O}$, all Δ-decompositions $\Sigma_1, \ldots, \Sigma_n$ of $\mathcal{O}$, and all Δ-encapsulating $\mathcal{L}$-realizations $\mathcal{O}_1, \ldots, \mathcal{O}_n$ and $\mathcal{O}'_1, \ldots, \mathcal{O}'_n$ of $\Sigma_1, \ldots, \Sigma_n$ for $\mathcal{O}$, we have $\mathcal{O}_i \equiv \mathcal{O}'_i$ for all $i \leq n$.

UDR thus guarantees that two realizations of the same Δ-decomposition are identical up to logical equivalence; the underlying decomposition does not need to be a finest or the finest. Konev et al. establish the *parallel interpolation property (PIP)*[7] as a sufficient condition for UDR which additionally ensures the existence of a unique finest Δ-decomposition in $\mathcal{L}$.

Theorem 3.12 (Konev et al., 2010)**.** *Let $\mathcal{L}$ be a fragment of SOL with PIP. Then the following hold.*

(1) $\mathcal{L}$-decompositions coincide with SOL-decompositions.

(2) $\mathcal{L}$ has UDR.

In particular, every $\mathcal{L}$-ontology $\mathcal{O}$ has a unique finest Δ-decomposition in $\mathcal{L}$.

Remark 3.13 (Konev et al., 2010). A number of DLs have PIP, including $\mathcal{EL}$, $\mathcal{ELH}$, $\mathcal{ALC}$, $\mathcal{ALCI}$, $\mathcal{ALCQ}$, $\mathcal{ALCQI}$. However, PIP fails already for the extension of $\mathcal{ALC}$ with role inclusions, and neither is (1) known for this DL. Although PIP can be restored for expressive DLs with role inclusions by including all role and individual names in Δ, this would not allow for the conclusion that they have UDR.

Modules and their properties. Analogously to Parikh's modules, we have to consider the $\mathcal{O}_i$ that realize a Δ-decomposition $\Sigma_1, \dots, \Sigma_n$ of $\mathcal{O}$ modulo logical difference. The $\mathcal{O}_i$ need not be modules: although the Σ_i are pairwise disjoint, the $\mathcal{O}_i$ can logically interact via Δ; hence they no longer encapsulate the knowledge about "their" Σ_i, unlike Parikh modules:

Example 3.14. Let $\mathcal{O} = \{A \sqsubseteq B,\ B \sqsubseteq B',\ B \sqsubseteq \neg B'\}$. If we take $\Delta = \{B\}$, then $\{A\}, \{B'\}$ is a Δ-decomposition, realized by $\mathcal{O}_1, \mathcal{O}_2$ with $\mathcal{O}_1 = \{A \sqsubseteq B\}$ and $\mathcal{O}_2 = \{B \sqsubseteq B',\ B \sqsubseteq \neg B'\}$. This realization is not Δ-encapsulating because $B \sqsubseteq \bot$ is entailed by $\mathcal{O}_2$, but not by $\mathcal{O}_1$. Hence $\mathcal{O}_1$ is not a logical module: $\mathcal{O} \models A \sqsubseteq \bot$, but $\mathcal{O}_1 \not\models A \sqsubseteq \bot$. If we replace $\mathcal{O}_1$ with $\mathcal{O}_1' = \mathcal{O}_1 \cup \{B \sqsubseteq \bot\}$, then $\mathcal{O}_1', \mathcal{O}_2$ is a Δ-encapsulating realization.

Δ-encapsulating $\mathcal{L}$-realizations ensure that all information about Δ is accessible in each single $\mathcal{O}_i$. Consequently, if $\mathcal{L}$ has UDR, then the $\mathcal{O}_i$ of a Δ-encapsulating $\mathcal{L}$-realizations are logical modules of $\mathcal{O}$ (Konev et al., 2010). It is reasonable to consider an additional module $\mathcal{O}_\Delta$ with $\widetilde{\mathcal{O}}_\Delta = \Delta$ and $\mathcal{O}_\Delta \equiv_\Delta^{\mathrm{dCE}} \mathcal{O}$, if it exists, and assume w.l.o.g. that $\mathcal{O}_\Delta$ is contained in all $\mathcal{O}_i$. In Example 3.14, a natural such $\mathcal{O}_\Delta$ would be $\{B \sqsubseteq \bot\}$. In case $\mathcal{O} \equiv_\Delta^{\mathrm{dCE}} \emptyset$, we have $\mathcal{O}_\Delta = \emptyset$.

We call the $\mathcal{O}_i$ and $\mathcal{O}_\Delta$ the Δ-*modules*, and their collection $\{\mathcal{O}_\Delta, \mathcal{O}_1, \dots, \mathcal{O}_n\}$ the Δ-*modularisation* of $\mathcal{O}$. Like Parikh modules, the $\mathcal{O}_i$ are not necessarily subsets of $\mathcal{O}$.

It has been observed by Konev et al. that, given UDR, each $\mathcal{O}_i$ provides coverage for Σ_i and is even self-contained. By definition, $\mathcal{O}_\Delta$ too provides coverage for Δ and is self-contained. In general, the $\mathcal{O}_i$ and $\mathcal{O}_\Delta$ are *not depleting* because they are not necessarily subsets of $\mathcal{O}$; see the remark in Section 3.1.

To measure *cohesion*, we revisit the ontologies from Examples 3.7 and 3.8.

7. For a detailed introduction and discussion of PIP as a property of a logic $\mathcal{L}$, we refer the reader to Konev et al. (2010).

Example 3.15. Consider the Zigzag_n family of ontologies from Example 3.8 for $n \geq 3$. If we choose $\Delta = \{A_i\}$ with $2 \leq i \leq n$, then the Δ-decomposition obtained consists of two parts: $\{A_1, \ldots, A_{i-1}, B_1, \ldots, B_{i-1}\}$ and $\{A_{i+1}, \ldots, A_{n+1}, B_i, \ldots, B_n\}$. Similarly, if $\Delta = \{B_i\}$ with $1 \leq i \leq n$, the corresponding Δ-decomposition is $\{A_1, \ldots, A_i, B_1, \ldots, B_{i-1}\}$ and $\{A_{i+1}, \ldots, A_{n+1}, B_{i+1}, \ldots, B_n\}$.

If we analyse all these cases, we notice that, for all $i \in \{1, \ldots, n\}$ the terms A_i and B_i are always either in the same part, or in related parts. The same happens for the terms A_{i+1} and B_i. In contrast with this situation, all the other pairs of terms can be separated by a suitable choice of Δ, so it is clear that the logical relation between these pairs is looser than the one between the pairs listed before. Hence, the Δ-decomposition reveals stronger logical relations between the terms of an ontology.

Example 3.16. Consider the ontology Tree from Example 3.7. For $\Delta = \{\mathtt{isMadeFrom}\}$ we get the decomposition $\Sigma_1 = \{\mathtt{Tree}, \mathtt{Seed}, \mathtt{Fruit}, \mathtt{Flower}, \mathtt{Branch}\}$ and $\Sigma_2 = \{\mathtt{Mountain}, \mathtt{Soil}\}$. It captures exactly the desired logical relation as described in Example 3.7.

Modular structure and its properties. Analogously to Parikh's approach, the modular structure can be based on a Δ-encapsulating $\mathcal{L}$-realization of the finest Δ-decomposition $\mathcal{O}_1, \ldots, \mathcal{O}_n$, plus the additional $\mathcal{O}_\Delta$, if it exists. In order for modules to be uniquely determined, we assume that $\mathcal{L}$ has PIP, although this is not a small restriction, given Remark 3.13. The basic Δ-modules are the $\mathcal{O}_i$ and $\mathcal{O}_\Delta$; the derived modules are their unions. As in Parikh's approach, Δ-modules and derived modules coincide, and the module generation mechanism is the same.

Because all modules contain $\mathcal{O}_\Delta$, they logically interact with each other unless $\mathcal{O} \equiv_\Delta^{\mathrm{dCE}} \emptyset$. However, since the $\mathcal{O}_i$ are Δ-inseparable and the Σ_i are pairwise disjoint, the only dependencies are between the $\mathcal{O}_i$ and the $\mathcal{O}_\Delta$, if it exists. More precisely, each $\mathcal{O}_i$ depends on $\mathcal{O}_\Delta$, but not vice versa. The dependency graph is thus a tree of depth 1 and width n, or it consists of n isolated nodes.

The choice of Δ strongly affects the granularity of the decomposition: In the extreme case $\Delta = \widetilde{\mathcal{O}}$, the set $\widetilde{\mathcal{O}} \setminus \Delta$ is empty and thus has only the empty partition.[8] We exclude this pathological case from the subsequent considerations. On the other extreme, if $\mathcal{O} \equiv_\Delta^{\mathrm{dCE}} \emptyset$ (including the case $\Delta = \emptyset$), then the unique finest Δ-decomposition coincides with the unique finest Parikh splitting, except for the additional $\mathcal{O}_\Delta$ in its realization. It seems reasonable to expect the following behaviour of the unique finest Δ-decomposition: (i) it is at least as fine as the unique finest Parikh splitting and (ii), which implies (i), the number of its parts increases monotonically with Δ. While this expected behaviour can be observed in Example 3.17, it does not occur in general, as Example 3.18 shows.

Example 3.17. The Δ-decomposition from Example 3.16 is finer than the Parikh splitting. If we increase Δ to, e.g., $\Delta' = \{\mathtt{isMadeFrom}, \mathtt{Tree}, \mathtt{Fruit}\}$, then the Δ-decomposition becomes even finer: $\Sigma_1 = \{\mathtt{Seed}\}$, $\Sigma_2 = \{\mathtt{Flower}, \mathtt{Branch}\}$, and $\Sigma_3 = \{\mathtt{Mountain}, \mathtt{Soil}\}$.

Example 3.18. Let $\mathcal{O}$ consist of two signature-disjoint axioms, such as $\mathcal{O} = \{A_1 \sqsubseteq B_1,\ A_2 \sqsubseteq B_2\}$. Then the unique finest Parikh splitting is Σ_1, Σ_2 with $\Sigma_i = \{A_i, B_i\}$, $i = 1, 2$, realized by $\mathcal{O}_1, \mathcal{O}_2$ with $\mathcal{O}_i = \{A_i \sqsubseteq B_i\}$, $i = 1, 2$. Also, Σ_1, Σ_2 is the unique finest Δ-decomposition for $\Delta = \emptyset$ and for $\Delta = \{A_1\}$ (or any other singleton Δ). If we increase Δ to $\Delta' = \{A_1, B_1\}$,

8. Note that partitions are usually defined to consist of non-empty parts.

then the part Σ_1 vanishes and Σ_2 remains, i.e., we get the singleton Δ'-decomposition consisting of Σ_2, realized by $\mathcal{O}_2$ (and $\mathcal{O}_\Delta = \mathcal{O}_1$).

Hence, Δ-decompositions are incomparable with Parikh splittings in general. In particular, because Property (ii) above fails, the selection of a suitable Δ in practice is non-trivial and constitutes the major challenge for Δ-decomposing an ontology. Konev et al. (2010) "do not expect signature decompositions to be a push-button technique, but rather envision an iterative and interactive process of understanding and improving the structure of an ontology, where the designer repeatedly chooses sets Δ and analyses the impact on the resulting decomposition". Hence, this method does not produce any stable, intrinsic modular structure—at least, we do not yet have conditions to ensure such a property.

Computational properties. In (Konev et al., 2010) the computability of signature Δ-decompositions has been analysed for those languages that have PIP. In particular, finding the finest Δ-decomposition is polynomial in DL-Lite, and in $\mathcal{EL}$ with further constraints (either $\Delta = \emptyset$ or $\mathcal{O}$ is role-acyclic, i.e., $\mathcal{O}$ has no entailments of the form $C \sqsubseteq \exists r_1. \cdots \exists r_n.C$ where C is an $\mathcal{EL}$-concept, the r_i are role names, and $n \geq 1$); ExpTime-complete in $\mathcal{ALC}, \mathcal{ALCI}, \mathcal{ALCQ}$ or $\mathcal{ALCQI}$, and in $\mathcal{ALCH}$ or $\mathcal{ALCHI}$ under the condition that Δ contains all role names from $\widetilde{\mathcal{O}}$. For languages without PIP the existence of a finest Δ-decomposition, and thus of a modular structure as described above, is not guaranteed.

3.3.3 Decomposition based on $\mathcal{E}$-connections

An orthogonal approach to decomposing an ontology into modules aims at identifying distinct subdomains present in an ontology, and making explicit how the ontology describes entities from one subdomain in terms of entities of others. For example, a medical ontology might deal with the non-overlapping subdomains of anatomy, diseases, and drugs, referring to anatomy concepts in order to define diseases and to concepts from both anatomy and diseases to define drugs.

In order to model non-overlapping subdomains and the logical interactions between them, the notion of an *$\mathcal{E}$-connection* has been defined (Kutz, Wolter, & Zakharyaschev, 2002; Kutz et al., 2004). $\mathcal{E}$-connections are based on a principled approach to combining several logics. To ensure that the combined formalism is computationally well-behaved, the constituent logics are required to fall under the general notion of an *abstract description system* (ADS), introduced by Baader, Lutz, Sturm, and Wolter (2002) to study *fusions* of logics. The semantics of fusions is designed to merge the involved interpretation subdomains, whereas that of $\mathcal{E}$-connection keeps them separate. ADSs include most of the commonly used DLs, but also modal and (propositional) temporal logics.

In the special case of DLs, an *$\mathcal{E}$-connection* is a set of ontologies—its *parts*—that are semantically interconnected via *link properties (LPs)*.[9] The intuitive idea is that every part covers a separate domain, and LPs allow to transfer information between these domains in the sense indicated above. More precisely, concepts from one application domain (e.g., in a medical ontology, certain drugs) can be defined in terms of concepts from other domains (e.g., certain diseases and body parts). To capture this intuition, the semantics of $\mathcal{E}$-connections is

9. Unlike other formalisms for decomposition, we omit here the formal definition of $\mathcal{E}$-connections since it is rather lengthy; we refer the interested reader to (Baader et al., 2002; Cuenca Grau et al., 2006; Jongebloed & Schneider, 2018).

based on DL interpretations whose domains are divided into pairwise disjoint subdomains. Given an $\mathcal{E}$-connection $\mathfrak{O}$ consisting of the parts $\mathcal{O}_1, \ldots, \mathcal{O}_n$, each $\mathcal{O}_i$ is associated with one of the subdomains; each concept name or individual in $\mathfrak{O}$ is associated with one $\mathcal{O}_i$ and interpreted within the respective subdomain; the same holds for some of the role names in $\mathfrak{O}$; each of the remaining role names in $\mathfrak{O}$ becomes a LP that is associated with a pair $(\mathcal{O}_i, \mathcal{O}_j)$ and interpreted as a binary relation between the respective subdomains. In the above example, three subdomains are needed, and the extensions of, e.g., the concept name `BodyPart` and the role name `hasLocation` are restricted to the anatomy subdomain. In contrast, the LP `affects` is interpreted as a binary relation that links the disease subdomain with the anatomy one.

In Cuenca Grau et al. (2006), a procedure for decomposing an ontology into an $\mathcal{E}$-connection is described. It yields a partition of the set of an ontology's axioms, together with connections between the parts, modelled as link properties. Despite being nondeterministic, the procedure yields the unique finest partition of the input ontology that is an $\mathcal{E}$-connection (where "refines" and "finest" are defined in the beginning of Section 3.3.1). Not every such partition is equivalent to the original ontology *under the $\mathcal{E}$-connection semantics*. In order to guarantee this kind of equivalence between the input and output ontology, it is necessary to assume that the input ontology is *safe*, which means that the satisfaction of its axioms does not depend on the existence of domain elements that do not participate in extensions of concept, role, or individual names. This notion is also known as *domain independence* in the database context (Abiteboul, Hull, & Vianu, 1995). For example, the axiom $\alpha = \neg A \sqsubseteq B$ is not safe because, if a model $\mathcal{I}$ of α is extended by a domain element that does not occur in the extension of any term, then α ceases to be satisfied.

Example 3.19. Consider the (safe) example ontology Koala2 given in Figure 1, created in the spirit of the toy ontology Koala.[10] Figure 2 provides the finest partition into an equivalent $\mathcal{E}$-connection. The nodes are labelled with the axioms contained in them, together with an informal name summarising their contents. The edges are labelled with the respective LP.

The $\mathcal{E}$-connection in Figure 2 comprises of two connected components: an isolated part containing only axiom α_{14} and the remainder of the ontology, decomposed into 5 parts. The largest part consists of the 6 axioms describing animals and their subconcepts, and the ranges of the `hasHabitat` and `hasGender` relations. These axioms cannot be further separated because of the fundamental property of $\mathcal{E}$-connections to be 'closed under subconcepts': e.g., due to the axiom $\texttt{Marsupial} \sqsubseteq \texttt{Animal}$, both concept names `Marsupial` and `Animal` must be associated with the same part and so must the respective axioms α_3 and α_4. The remaining parts of this component contain the axioms describing habitats (including forests), climate, vegetation, and genders. The edges and their respective LPs capture the logical interaction between the axioms in the ontology in an intuitive way: animals are described in terms of their habitat and gender; habitats are described in terms of their climate and vegetation; climate and vegetation affect each other. Finally, the isolated part correctly represents the fact that α_{14} does not logically interact with the remainder of the ontology.

In general, every $\mathcal{E}$-connection $\mathfrak{O} = (\mathcal{O}_1, \ldots, \mathcal{O}_n)$ of an ontology $\mathcal{O}$ induces a directed graph $G_{\mathfrak{O}}$—called the *$\mathcal{E}$-connection graph*—whose nodes are the $\mathcal{O}_i$ and whose edges are determined by the LPs. An edge from $\mathcal{O}_i$ to $\mathcal{O}_j$ indicates that $\mathcal{O}_i$ imports vocabulary from

10. `http://protege.stanford.edu/plugins/owl/owl-library/koala.owl`

$$
\begin{aligned}
\mathsf{Koala2} = \{\ \alpha_1\colon\ & \texttt{Koala} \sqsubseteq \texttt{Marsupial} \sqcap \exists\,\texttt{hasHabitat}.\texttt{DryEucalyptForest},\\
\alpha_2\colon\ & \texttt{Quokka} \sqsubseteq \texttt{Marsupial},\\
\alpha_3\colon\ & \texttt{Marsupial} \sqsubseteq \texttt{Animal},\\
\alpha_4\colon\ & \texttt{Animal} \sqsubseteq \exists\,\texttt{hasGender}.\top \sqcap \exists\,\texttt{hasHabitat}.\top,\\
\alpha_5\colon\ & \top \sqsubseteq \forall\,\texttt{hasHabitat}.\texttt{Habitat},\\
\alpha_6\colon\ & \top \sqsubseteq \forall\,\texttt{hasGender}.\texttt{Gender},\\
\alpha_7\colon\ & \texttt{DryEucalyptForest} \sqsubseteq \texttt{Forest},\\
\alpha_8\colon\ & \texttt{Forest} \sqsubseteq \texttt{Habitat},\\
\alpha_9\colon\ & \texttt{Habitat} \sqsubseteq \exists\,\texttt{hasClimate}.\texttt{Climate} \sqcap \exists\,\texttt{hasVegetation}.\texttt{Vegetation},\\
\alpha_{10}\colon\ & \texttt{Climate} \sqsubseteq \exists\,\texttt{controls}.\texttt{Vegetation},\\
\alpha_{11}\colon\ & \texttt{Vegetation} \sqsubseteq \exists\,\texttt{dependsOn}.\texttt{Climate},\\
\alpha_{12}\colon\ & \texttt{Gender}(\texttt{female}),\\
\alpha_{13}\colon\ & \texttt{Gender}(\texttt{male}),\\
\alpha_{14}\colon\ & \texttt{Continent}(\texttt{australia})\ \}
\end{aligned}
$$

Figure 1: Toy ontology Koala2.

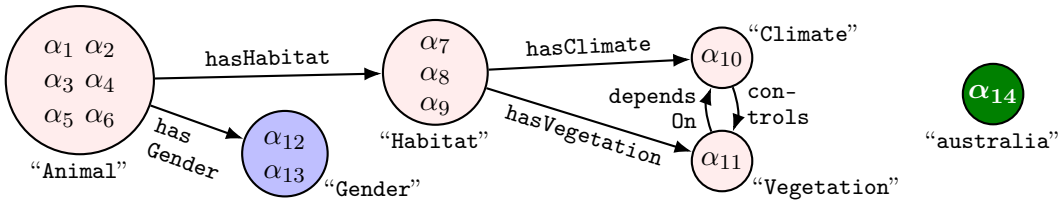


Figure 2: $\mathcal{E}$-connection resulting from partitioning the ontology Koala2.

$\mathcal{O}_j$, i.e., the meaning of the concepts in $\mathcal{O}_i$ depends on that in $\mathcal{O}_j$. Based on this "imports" relation, the $\mathcal{O}_i$ can be divided into three types:

- *light/red:* parts with outgoing edges ("importing parts"),
- *medium/blue:* parts with incoming but no outgoing edges ("imported parts"), and
- *dark/green:* isolated parts (they indicate subdomains unrelated to the rest of the ontology and could be maintained in a separate ontology).

Unlike the two approaches described previously, decomposition based on $\mathcal{E}$-connections splits the set of the input ontology's *axioms*. As a consequence, two equivalent ontologies may have different $\mathcal{E}$-connections, as we can see in the next example.

Example 3.20. Consider the ontology $\mathcal{O} = \{A_1 \sqsubseteq B_1,\ A_2 \sqsubseteq B_2\}$: its $\mathcal{E}$-connection has two parts with a single axiom each. Now $\mathcal{O}$ is equivalent to $\mathcal{O}' = \{\top \sqsubseteq (\neg A_1 \sqcup B_1) \sqcap (\neg A_2 \sqcup B_2)\}$ whose $\mathcal{E}$-connection only has a single part, $\mathcal{O}'$ itself.

Although we can associate a partition of the signature of an ontology with an $\mathcal{E}$-connection, different parts of an $\mathcal{E}$-connection can share terms, i.e., the set of terms associated with a part is only a subset of its signature; we will discuss this in the following. Furthermore, the process of partitioning is completely automatic.

Modules and their properties. A single part of an $\mathcal{E}$-connection is not necessarily logically closed: its outgoing LPs point to parts on which it logically depends. In the initial example, the "drugs" part depends on the other two parts, and the "diseases" part depends on the "anatomy" part. The modules of an $\mathcal{E}$-connection are thus obtained by following the logical dependencies, modelled by the LPs. More precisely, in an $\mathcal{E}$-connection $\mathfrak{O} = (\mathcal{O}_1, \dots, \mathcal{O}_n)$ of an ontology $\mathcal{O}$, each sub-collection $(\mathcal{O}_{i_1}, \dots, \mathcal{O}_{i_m})$ of its parts determines an *$\mathcal{E}$-module* $\mathcal{M}_{i_1,\dots,i_n}$, which is the union of all the $\mathcal{O}_{i_j}$ and all $\mathcal{O}_j$ that are reachable from some $\mathcal{O}_{i_j}$ in $G_{\mathfrak{O}}$ via some path of arbitrary length (if nominals occur, then predecessors in $G_{\mathfrak{O}}$ need to be taken into account as well).

Koala2 (Figure 2) has 13 $\mathcal{E}$-modules: (1) `Gender`, (2) `Climate`∪`Vegetation`, (3) `Habitat`∪ `Climate` ∪ `Vegetation`, (4) the union of 1 and 2 with `Animal`, (5) `australia`, and (6–13) any union of two or more of 1–5.

Cuenca Grau et al. (2006) show that all $\mathcal{E}$-modules provide *coverage* and are *self-contained*. The property of being depleting is not discussed; however, every $\mathcal{E}$-module $\mathcal{M}_i$ determined by a single part $\mathcal{O}_i$ is shown to satisfy additional, similar encapsulation properties:

(i) For every concept name A associated with $\mathcal{O}_i$, the module $\mathcal{M}_i$ entails the same subsumptions $A \sqsubseteq B$ and $B \sqsubseteq A$ as $\mathcal{O}$, where B is an *arbitrary* concept name from $\mathcal{O}$.

(ii) $\mathcal{O}$ does not entail any subsumptions between concept names occurring in $\mathcal{M}_i$ and concept names not occurring in $\mathcal{M}_i$.

Unlike in the previous two approaches, all $\mathcal{O}_i$, and thus all modules, are subsets of $\mathcal{O}$. Even though every concept and role name is associated with exactly one $\mathcal{O}_i$ (or one pair), the signatures of the $\mathcal{O}_i$ are typically not disjoint.

The *cohesion* of an $\mathcal{E}$-module can vary greatly from ontology to ontology. Due to conditions (i) and (ii), the size of the partition and thus the number of basic modules is bounded by the number of top-level concepts in the original ontology.[11] Consequently, $\mathcal{E}$-modules of some ontologies combine axioms that are related not logically but because they involve sub-concepts of the same (generic) top-level concept.

Modular structure and its properties. The basic modules are those $\mathcal{E}$-modules $\mathcal{M}_i$ that are determined by a single part $\mathcal{O}_i$; the derived modules are their unions. As in the previous two approaches, $\mathcal{E}$-modules $\mathcal{M}_{i_1,\dots,i_n}$ and derived modules coincide, and the generation mechanism is the same.

The edges of the $\mathcal{E}$-connection graph $G_{\mathfrak{O}}$ induce a dependency relation between the parts $\mathcal{O}_i$ of the partition: if there is a path from $\mathcal{O}_i$ to $\mathcal{O}_j$ in $G_{\mathfrak{O}}$, then the following property holds.

$$\text{For every basic module } \mathcal{M}_k, \text{ if } \mathcal{O}_i \subseteq \mathcal{M}_k, \text{ then } \mathcal{O}_j \subseteq \mathcal{M}_k.$$

In other words, the axioms in $\mathcal{O}_i$ logically depend on the axioms in $\mathcal{O}_j$ because the former need the latter to form modules that encapsulate knowledge—but not necessarily the other way round.

Dependencies between the parts imply inclusions between basic modules: if $\mathcal{O}_i, \mathcal{O}_j$ are part of a cycle, then $\mathcal{M}_i = \mathcal{M}_j$; if there is a path from $\mathcal{O}_i$ to $\mathcal{O}_j$ but not vice versa, then $\mathcal{M}_j \subsetneq \mathcal{M}_i$; otherwise $\mathcal{M}_i, \mathcal{M}_j$ are orthogonal (but not necessarily disjoint). In the case of

11. The same holds with respect to the bottom-level concepts, but their number is usually much larger.

Koala2 (Figure 2) the modules determined by `Climate` and `Vegetation` coincide, and are strictly contained in the module for `Habitat`.

$\mathcal{E}$-connections capture a further type of logical dependency between their modules: the overlapping of two distinct $\mathcal{M}_i, \mathcal{M}_j$. In this case, the two modules share one or more parts $\mathcal{O}_k$. This is represented in $G_{\mathfrak{D}}$ by the $\mathcal{O}_k$ being reachable from both $\mathcal{O}_i$ and $\mathcal{O}_j$.

The granularity of the modular structure induced by an $\mathcal{E}$-connection can vary greatly from ontology to ontology, see the observations related to cohesion above. As a consequence, some ontologies do not decompose well into an $\mathcal{E}$-connection due to certain modelling practices:

Example 3.21. Robert Stevens' Periodic Table Ontology Periodic,[12] which describes the elements of the periodic table and their properties, has a single top-level concept `DomainEntity`, and thus the finest $\mathcal{E}$-connection consists of a single node containing all axioms. In contrast, the domain modelled by Periodic is naturally partitioned and the ontology expresses this separation.

Example 3.22. The finest $\mathcal{E}$-connection for each of the ontologies Tree and Zigzag_n from Examples 3.7 and 3.8 consists of a single node, too (for all n). The reason for Tree is that all axioms share the role `isMadeFrom`, and among the first five axioms, each two consecutive axioms share a concept, which prevents the role `isMadeFrom` from being an LP. The reason for Zigzag_n is that for each $i \in \{1, \dots, n\}$, both A_i and B_i belong to the same part since $\mathsf{Zigzag}_n \models A_i \sqsubseteq B_i$. The same holds for A_{i+1} and B_i.

The other extreme is possible, too: it is easy to handcraft ontologies where every node in the $\mathcal{E}$-connection graph consists of exactly one axiom. Furthermore, $\mathcal{E}$-connections are incomparable in terms of granularity with Parikh splittings:

Example 3.23. Consider again the ontologies $\mathcal{O}$ and $\mathcal{O}'$ from Example 3.20: both the finest Parikh splitting and the finest $\mathcal{E}$-connection of $\mathcal{O}$ have two parts, one for each axiom. The equivalent ontology $\mathcal{O}'$ cannot be split by $\mathcal{E}$-connections, whereas the finest Parikh splitting remains the same. This is due to the fact that every part of an $\mathcal{E}$-connection must be a subset of the original ontology, whereas a realization of a Parikh splitting does not need to consist of subsets.

However, if one considers a restricted variant of Parikh splittings that admits only realisations that partition the original ontology $\mathcal{O}$, then the finest Parikh splitting of $\mathcal{O}$ always yields an $\mathcal{E}$-connection: since the n parts of its realisation are pairwise signature-disjoint and subsets of $\mathcal{O}$, they constitute an $\mathcal{E}$-connection with n isolated parts. Consequently, the finest $\mathcal{E}$-connection is at least as fine as the finest Parikh splitting (in this variant).

Computational properties. The finest $\mathcal{E}$-connection for a given ontology $\mathcal{O}$ in DLs up to $\mathcal{SHOIQ}$ can be computed using Cuenca Grau et al.'s (2006) algorithm in time polynomial in the size of $\mathcal{O}$. This algorithm only guarantees structural compatibility of the resulting $\mathcal{E}$-connection with $\mathcal{O}$ and not logical equivalence under the $\mathcal{E}$-connection semantics (see above). The safety check can be performed in polynomial time as a preprocessing step. Altogether every safe $\mathcal{SHOIQ}$ ontology can be partitioned into a finest logically equivalent $\mathcal{E}$-connection in polynomial time. Jongebloed and Schneider (2018) give a linear-time algorithm for the same task in DLs up to $\mathcal{SROIQ}$. They have also extended the safety check to $\mathcal{SROIQ}$ minus the universal role, and argued why the latter is a strict boundary to the safety check.

12. `http://www.cs.man.ac.uk/~stevensr/ontology/periodic.zip`

In the meantime, a prototype has been implemented and tested in Jongebloed's Master's thesis (in preparation). The performance turns out to be outstanding, but most of the observed $\mathcal{E}$-connections are discouragingly coarse-grained. Additional heuristics are needed to find out whether this approach is useful at all.

3.3.4 Early attempts to identify a modular structure for Σ-modules

The extraction of a module $\mathcal{M}$ for a given signature Σ from an ontology $\mathcal{O}$ is a well-studied task, see Section 2 (Konev et al., 2008, 2009; Cuenca Grau et al., 2009; Sattler et al., 2009). R_Σ-modules, for a fixed inseparability relation R (see Section 3.2), are designed to preserve the entailments (if R is defined deductively) or the models (if R is defined model-theoretically) over a given signature Σ and, because of self-containment, [13] over the signature $\widetilde{\mathcal{M}}$ of $\mathcal{M}$ itself. However, $\mathcal{M}$ does *not* guarantee to preserve any entailments between the terms in $\Sigma \cup \widetilde{\mathcal{M}}$ and the remaining terms in $\mathcal{O}$; it therefore does not capture any information on the logical relationship between the terms in $\mathcal{M}$ and those outside. This observation makes the determination of the "signature of interest" Σ more difficult than one might expect, and as long as a user is unsure about the signature to specify, the ability to extract a single module is of little help to them.

As a simple (abstract) example, suppose an ontology user or engineer wants to extract from an ontology $\mathcal{O}$ a module $\mathcal{M}$ that captures how a concept name A is described in $\mathcal{O}$, but they do not know which other terms in $\mathcal{O}$ are related to A. The most obvious choice for a signature Σ that determines $\mathcal{M}$ would be $\Sigma = \{A\}$. However, for most "sound" ontologies, $\mathcal{M} = \emptyset$ will be a module for Σ because the only interesting non-tautologous entailments that can be formulated using only A are $A \sqsubseteq \bot$ and $\top \sqsubseteq A$—and they are both unlikely to follow from $\mathcal{O}$ (they mean that A is unsatisfiable w.r.t. $\mathcal{O}$ or a top-level concept). Indeed, if we consider the ontology Tree from Example 3.7 and are interested in the concept name $\mathtt{Tree}$, then the empty ontology is a module of Tree for Σ. One might argue that existing module-extraction approaches for expressive ontology languages, including locality-based modules, do not guarantee to generate a minimal module. However, it would certainly not help to try and rely on modules containing "a bit more" than they should. In the case of Tree, the $\top\bot^*$-module for Σ is indeed empty.

If we want to support our ontology user/editor in finding a module (for A) that is appropriate for their needs, we could try providing them with a list of *all* possible modules of $\mathcal{O}$ (containing) to choose from. However, this naïve approach is infeasible as $\mathcal{O}$ may have exponentially many modules:

Example 3.24. Let us consider the ontology $\mathsf{All} = \{A_1 \sqsubseteq B_1,\ A_2 \sqsubseteq B_2,\ \ldots,\ A_n \sqsubseteq B_n\}$. Then every subset of All is a $\top\bot^*$-module for some signature $\Sigma \subseteq \widetilde{\mathsf{All}}$.

One might hope that the exponential behaviour observed for this artificial ontology does not tend to occur for "real-life" ontologies. This hope was had by Parsia and Schneider (2010), who attempted to compute a modular structure for an ontology based on the set of *all* its R_Σ-modules, based on a fixed inseparability relation R and module *extraction* approach x (in particular, locality-based modules). Before coming back to these early attempts, we will first discuss the resulting modular structures and the requirements to the module notion x.

13. See Section 3.2 for a summary of the core assumptions regarding modules.

Modules and their properties. We consider an inseparability relation R and a module notion x which, for every $\mathcal{O}$ and Σ, yields an R_Σ-module x-mod$(\Sigma, \mathcal{O})$ for Σ in $\mathcal{O}$ that is

(M0) self-contained,

(M1) uniquely determined, and

(M2) a subset of $\mathcal{O}$.

Property (M0) is advisable given the discussion about self-containment and interfaces in Section 3.1, and (M1) and (M2) reflect the fact that module *extraction* approaches are to be used, which typically yield uniquely determined subsets.[14] It should be noted that *coverage* is provided by R_Σ-modules by definition; we do not require that they are *depleting*.

The module notion x does not need to have been designed with a notion of a modular structure or basic and derived modules in mind. As a prominent example for x, we will often refer to locality-based modules (Section 2) throughout this section, in particular $\top\bot^*$-modules: they comprise perhaps the most widely known language-independent and computationally well-behaved module notion, and they are readily usable via the OWL API.

As explained in the last paragraph of Section 2, all module notions satisfying (M2) are necessarily syntax dependent, i.e., two equivalent ontologies can lead to two different, non-equivalent modules for the same seed signature and thus to different modular structures.

Example 3.25. Consider the ontology $\mathcal{O} = \{A_1 \sqsubseteq A_2,\ A_2 \sqsubseteq A_3,\ A_3 \sqsubseteq A_4\}$ and let $\Sigma_1 = \{A_1, A_2\}$, $\Sigma_2 = \{A_3, A_4\}$, and $\mathcal{M}_i = x\text{-mod}(\Sigma_i, \mathcal{O})$ for $i = 1, 2$. If x is $\top\bot^*$, then $\mathcal{M}_1 = \top\bot^*\text{-mod}(\Sigma_1, \mathcal{O}) = \{A_1 \sqsubseteq A_2\}$ and $\mathcal{M}_2 = \top\bot^*\text{-mod}(\Sigma_2, \mathcal{O}) = \{A_3 \sqsubseteq A_4\}$. However, $\mathcal{M} = \mathcal{M}_1 \cup \mathcal{M}_2$ is not a module: its signature contains both A_2 and A_3, but $\mathcal{M}$ does not contain the second axiom, thus violating self-containment.

Hence, if we were to define the derived modules to be exactly those that are the union of two basic modules, then (a) the distinction between basic and derived modules would be arbitrary and unintuitive (there could be very large basic modules containing derived modules), and (b) "union" would not be a meaningful derivation mechanism because it sometimes produces a module and sometimes it does not.

The last problem could be repaired by considering the following derivation mechanism: given modules $\mathcal{M}_1, \ldots, \mathcal{M}_n$ of an ontology $\mathcal{O}$, the module derived from $\mathcal{M}_1, \ldots, \mathcal{M}_n$ is x-mod$(\Sigma, \mathcal{O})$ where Σ is union of the signatures of the $\mathcal{M}_i$. One could then define derived modules to be exactly those that can be obtained this way; all other modules would be basic. However, this choice is unintuitive, too: (a) determining whether a module $\mathcal{M}$ is basic or derived would be highly complex because one would have to check $\mathcal{M}$ against every union over a subset $\{\mathcal{M}_1, \ldots, \mathcal{M}_n\}$ of the set of all modules; (b) an ontology $\mathcal{O}$ could have a chain $\mathcal{M}_1 \subseteq \cdots \subseteq \mathcal{M}_n$ of basic modules, e.g., when the signatures of $\mathcal{O}$'s axioms are monotonously increasing w.r.t. the subset relation.

Cohesion may vary greatly, depending on the module notion x. In the worst case, x can be such that x-mod$(\Sigma, \mathcal{O}) = \mathcal{O}$ for every signature Σ. Then obviously the only module $\mathcal{O}$ is of the lowest possible cohesion; see, e.g., the discussion in the previous subsection where Tree did not decompose. In the best case, take x to be the minimal R_Σ-module for

14. The so-called *projection* and *subsumption modules* (Chen, Ludwig, & Walther, 2018; Chen, Ludwig, Ma, & Walther, 2019b) are not unique, thus do not satisfy (M1) and are therefore not further discussed here.

every Σ.[15] Then the basic modules are of very high cohesion: two axioms occurring in a $\subseteq$-minimal module $\mathcal{M}$ are strongly related in a logical sense because separating them is likely to compromise entailments over the signature covered by $\mathcal{M}$.

Modular structure and its properties. In order to obtain a modular structure from a family of x-modules, we denote with $\mathfrak{F}^x(\mathcal{O})$ the set of all x-modules in $\mathcal{O}$. When no ambiguity may arise, we drop x and write $\mathfrak{F}(\mathcal{O})$. We set $\mathfrak{G} = \mathfrak{F}^x(\mathcal{O})$ and $\rightarrow$ to be the set inclusion relation $\subseteq$. Under certain assumptions to the module notion x, this dependency relation can be represented by the complete join-semilattice whose 1-element is $\mathcal{O}$, whose 0-element is the module for $\Sigma = \emptyset$,[16] and where the join of any two modules $\mathcal{M}_1$ and $\mathcal{M}_2$ is the smallest module containing both $\mathcal{M}_1$ and $\mathcal{M}_2$. For the latter to be well-defined, we need to require not only Properties (M1)–(M2) above but also the following two.

(M3) The function $x\text{-mod}(\cdot,\cdot)$ is monotonic in the first argument, i.e., if $\Sigma \subseteq \Sigma'$, then $x\text{-mod}(\Sigma, \mathcal{O}) \subseteq x\text{-mod}(\Sigma', \mathcal{O})$.

(M4) For all x-modules $\mathcal{M}$ of $\mathcal{O}$, it holds that $\mathcal{M} = x\text{-mod}(\widetilde{\mathcal{M}}, \mathcal{O})$.

Property (M4) is the 'operational' variant of the assumption that R_Σ-modules are self-contained, as discussed at the beginning of Section 3.2. Although x-modules do not need to be minimal, (M4) ensures that $\mathcal{M}$ covers $\mathcal{O}$ for its own signature but not more. Interestingly, (M4) does not follow from self-containment and depletingness of x-modules (see Def. 2.4) since the module extraction function $x\text{-mod}(\cdot,\cdot)$ could add different "extraneous" axioms depending on the first parameter. All known variants of locality-based modules (whether syntactic or semantic, top or bottom, nested or unnested), however, satisfy (M1)–(M4) (Konev et al., 2009; Sattler et al., 2009).

If the module notion x satisfies properties (M1)–(M4), then for all modules $\mathcal{M}_1, \mathcal{M}_2$ of $\mathcal{O}$ there is a unique $\subseteq$-minimal x-module of $\mathcal{O}$ that contains both $\mathcal{M}_1, \mathcal{M}_2$, namely $x\text{-mod}(\widetilde{\mathcal{M}_1} \cup \widetilde{\mathcal{M}_2}, \mathcal{O})$.

Unlike in the previous approaches, $\mathfrak{G}$ does not partition the input ontology; the modules in $\mathfrak{G}$ typically overlap.

The *granularity* can vary greatly depending on the module notion x and the ontology $\mathcal{O}$, analogously to the variability of cohesion: if x generates only large modules from $\mathcal{O}$, then the resulting modular structure $(\mathfrak{F}^x(\mathcal{O}), \subseteq)$ will be coarse-grained; if x generates smaller modules, then the granularity of $(\mathfrak{F}^x(\mathcal{O}), \subseteq)$ will be finer.

Computational properties. As already seen in Example 3.24, every subset of an ontology $\mathcal{O}$ may be an x-module for some signature. Thus, even when x-modules can be computed efficiently (as is the case for MEX or locality-based modules), the size of the modular structure is exponential in the worst case. Our previous study (Parsia & Schneider, 2010) shows that the situation is not significantly different for existing ontologies: the experiment on a number of well-designed and sufficiently diverse ontologies shows that (a) the number of modules is so large that a complete extraction is possible only for ontologies with up to ≈ 50 axioms; and (b) for a fixed ontology, the number of modules extracted from a sub-ontology grows

15. In this case, we need to use a suitable notion of minimality to ensure uniqueness (M1).
16. Note that the module $\text{mod}(\emptyset, \mathcal{O})$ is *not* necessarily empty. For example, the axiom $A \sqcup \neg B \sqsubseteq A \sqcap \neg B$ is neither $\bot$- nor $\top$-local w.r.t. *any* signature.

exponentially with the size of the sub-ontology. The latter suggests that the number of modules of an ontology indeed grows exponentially with the size of the ontology.

Even attempts to reduce the prohibitively large set of R-modules to a reasonable subset did not help: we observed (Parsia & Schneider, 2010) that a large proportion of the modules is of distinctively low cohesion, called "fake", but the number of the remaining "genuine" modules still exhibits an exponential growth, too. A "fake" module falls apart into smaller modules which are logically independent because their signatures are (almost) disjoint—a predecessor variant of the notion of genuine and fake modules introduced in Section 4.

4. Atomic Decomposition

In the previous section, we have seen a variety of existing approaches to decompose an ontology resulting in different kinds of modular structures with different properties–all of which provide strong logical guarantees. None of these approaches, however, results in a modular structure that would be suitable for modular ontology engineering, i.e., one that has a suitable number of basic modules with strong cohesion, a dependency relation which allows us to work with modules of a varying granularity, a suitable module generation mechanism, and whose computation is of acceptable complexity.

In this section, we assume we have fixed a notion of modules where each module is unique for a given seed signature, a subset of the ontology, and depleting. We introduce a new modular structure based on the notion of a *genuine module*, i.e., a module that cannot be decomposed into smaller modules. In this sense, genuine modules are the basic modules of our structure and of all derived modules.

The new modular structure will be based on two fundamental notions: (a) an *atom* of the input ontology $\mathcal{O}$ is a maximal subset of axioms that are never separated by any module of $\mathcal{O}$. Consequently, $\mathcal{O}$'s atoms partition $\mathcal{O}$ into highly cohesive subsets, and the number of atoms is bounded by the number of $\mathcal{O}$'s axioms. (b) The *dependency relation* between atoms of $\mathcal{O}$ captures a further kind of cohesion and allows for a natural definition of basic modules of $\mathcal{O}$. We will see that both the set of atoms and the dependency relation can be computed in quadratic time with a linear number of calls to a subroutine extracting x-modules.

Before we define these new terms, we clarify the requirements to the underlying module notion x.

4.1 Requirements

From now on, we fix an inseparability relation R and a module notion x which, for any given ontology $\mathcal{O}$ and signature Σ, yields an R_Σ-module $x\text{-mod}(\Sigma, \mathcal{O})$. The function $x\text{-mod}(\cdot, \cdot)$ must satisfy the following properties; the first four have already been fixed in Section 3.3.4 but we repeat them for readability.

(M0) $x\text{-mod}(\Sigma, \mathcal{O})$ is a self-contained R_Σ-module of $\mathcal{O}$.

(M1) $x\text{-mod}(\Sigma, \mathcal{O})$ is uniquely determined.

(M2) $x\text{-mod}(\Sigma, \mathcal{O})$ is a subset of $\mathcal{O}$.

(M3) The function $x\text{-mod}(\cdot, \cdot)$ is monotonic in the first argument.

(M4) For all x-modules $\mathcal{M}$ of $\mathcal{O}$, it holds that $\mathcal{M} = x\text{-mod}(\widetilde{\mathcal{M}}, \mathcal{O})$.

(M5) $x\text{-mod}(\Sigma, \mathcal{O})$ is a depleting R_Σ-module of $\mathcal{O}$ (see Def. 2.4).

(M6) For every tautology $\alpha \in x\text{-mod}(\Sigma, \mathcal{O})$, we have $\alpha \in x\text{-mod}(\Sigma \cap \widetilde{\alpha}, \mathcal{O})$.

Property (M6) is a technical subtlety that is required later only in the special case of α being a tautology: in this case, α does not need to be part of any module by definition of inseparability or even depletingness; however, since x-modules are not required to be minimal, for some seed signatures Σ they may contain α, and for others they may not. (M6) makes the natural requirement that the membership of any "empty" axiom α in the x-module for Σ does not depend on the terms in Σ that do not occur in α. For example, if $\alpha = A \sqsubseteq A \sqcup B$ and $\alpha \notin x\text{-mod}(\{A, B\}, \mathcal{O})$, then we cannot have, e.g., $\alpha \in x\text{-mod}(\{A, B, C\}, \mathcal{O})$.

Properties (M0)–(M6) are satisfied by locality-based and MEX modules; the suitability of the remaining module notions mentioned in Section 2.3 is discussed in Section 4.5. All of the following definitions and theoretical results apply to any module notion x with these properties. Only 'soft' parameters such as cohesion and granularity, as well as computational properties depend on the choice of x. We will get back to this last observation in Section 4.4; in the meantime we will often omit x when no confusion can arise.

We continue to use the notion $\mathfrak{F}^x(\mathcal{O}) = \{x\text{-mod}(\Sigma, \mathcal{O}) \mid \Sigma \subseteq \widetilde{\mathcal{O}}\}$ from Section 3.3.4 and write $\mathfrak{F}(\mathcal{O})$ when x is clear from the context or we do not have a specific x in mind.

4.2 Atoms, Dependency, and the Atomic Decomposition

Atoms are defined via the equivalence relation $\sim_\mathcal{O}$ which captures that axioms always co-occur in modules.

Definition 4.1. Let $\mathcal{O}$ be an ontology and $\alpha, \beta \in \mathcal{O}$. Then $\alpha \sim_\mathcal{O} \beta$ if, for all $\mathcal{M} \in \mathfrak{F}(\mathcal{O})$, we have $\alpha \in \mathcal{M}$ if and only if $\beta \in \mathcal{M}$.

It is immediate that $\sim_\mathcal{O}$ is indeed an equivalence relation.

Atoms are maximal subsets of axioms that are not separated by any module.

Definition 4.2. The *atoms* of an ontology $\mathcal{O}$ are the equivalence classes of $\sim_\mathcal{O}$. The set of atoms of $\mathcal{O}$ is denoted by $\mathfrak{A}(\mathcal{O})$.

We denote atoms by $\mathfrak{a}, \mathfrak{b}, \ldots$ The following observation is immediate.

Observation 4.3. *For every ontology $\mathcal{O}$, the following hold.*

1. *$\mathfrak{A}(\mathcal{O})$ is a partition of $\mathcal{O}$.*
2. *$\#\mathfrak{A}(\mathcal{O}) \leq \#\mathcal{O}$ (i.e., the number of atoms is bounded by the number of axioms in $\mathcal{O}$).*
3. *Every module $\mathcal{M} \in \mathfrak{F}(\mathcal{O})$ is a disjoint union of atoms.*

While every module of $\mathcal{O}$ is partitioned into one or several atoms, it is not necessarily the case that every atom occurs in some module, e.g., some tautologies do not occur in the x-module for any signature; we can think of these as obvious tautologies. If $\mathcal{O}$ contains such axioms, then by definition they constitute a single atom, which we call $\mathfrak{t}$; otherwise $\mathfrak{t}$ is empty

and thus not an atom. To make the following considerations more readable, and without loss of generality, we assume that $\mathfrak{t} = \emptyset$; we can always employ some preprocessing step that identifies $\mathfrak{t}$ and removes it from $\mathcal{O}$, resulting in an equivalent ontology.

Next, we define a binary relation $\succeq$ between atoms that captures the dependencies between atoms in terms of co-occurrence in modules: if $\mathfrak{a} \succeq \mathfrak{b}$, then every module that contains $\mathfrak{a}$ must contain $\mathfrak{b}$ too.

Definition 4.4. Let $\mathfrak{a}, \mathfrak{b}$ be atoms of an ontology $\mathcal{O}$. We say that $\mathfrak{a}$ *depends on* $\mathfrak{b}$ (written $\mathfrak{a} \succeq \mathfrak{b}$ or $\mathfrak{b} \preceq \mathfrak{a}$) if, for every $\mathcal{M} \in \mathfrak{F}(\mathcal{O})$, $\mathfrak{a} \subseteq \mathcal{M}$ implies $\mathfrak{b} \subseteq \mathcal{M}$. The relation $\succeq$ is called the *dependency relation* of $\mathcal{O}$.

The relation $\succeq$ is obviously a partial order: reflexivity and transitivity are immediate, and antisymmetry follows from the observation that, if both $\mathfrak{a} \succeq \mathfrak{b}$ and $\mathfrak{b} \succeq \mathfrak{a}$ hold, then $\alpha \sim_{\mathcal{O}} \beta$ for all axioms $\alpha \in \mathfrak{a}$ and $\beta \in \mathfrak{b}$; consequently $\mathfrak{a} = \mathfrak{b}$.

As usual, we use $\succ$ for the strict partial order underlying $\succeq$, i.e., $\mathfrak{a} \succ \mathfrak{b}$ if $\mathfrak{a} \succeq \mathfrak{b}$ and $\mathfrak{a} \neq \mathfrak{b}$. Since $(\mathfrak{A}(\mathcal{O}), \succ)$ is a *strict poset* (partially ordered set), we can represent it by means of a *Hasse diagram*. Furthermore, we can make use of the standard notions of a principal ideal[17] and an antichain:

Definition 4.5. Let $\mathcal{O}$ be an ontology.

1. The poset $(\mathfrak{A}(\mathcal{O}), \succ)$ is called the *atomic decomposition (AD)* of $\mathcal{O}$.
2. The *principal ideal* of an atom $\mathfrak{a} \in \mathfrak{A}(\mathcal{O})$ is the set $\downarrow\mathfrak{a} := \bigcup\{\mathfrak{b} \mid \mathfrak{a} \succeq \mathfrak{b}\}$.
3. An *antichain* of atoms is a set $\mathfrak{B} \subseteq \mathfrak{A}(\mathcal{O})$ such that $\mathfrak{a} \not\succeq \mathfrak{b}$ for any two distinct $\mathfrak{a}, \mathfrak{b} \in \mathfrak{B}$.

In case the module notion matters, we will explicitly mention x and speak of the *x-atomic decomposition* of $\mathcal{O}$, for short *x-AD*, denoted $(\mathfrak{A}^x(\mathcal{O}), \succ_x)$.

Example 4.6. Let us consider the Dog ontology defined as follows.

$$\begin{aligned} \mathsf{Dog} = \{\ \alpha_1 &: \mathtt{Poodle} \sqsubseteq \mathtt{Dog} \sqcap \exists \mathtt{hasCoat}.\mathtt{Curly}, \\ \alpha_2 &: \quad \mathtt{Dog} \sqsubseteq \mathtt{Mammal} \sqcap \exists \mathtt{hasPart}.\mathtt{Tail}, \\ \alpha_3 &: \mathtt{Mammal} \sqsubseteq \mathtt{Animal} \sqcap \exists \mathtt{hasPart}.\mathtt{Hair}\ \} \end{aligned}$$

The three "popular" notions of locality $\bot, \top$, and $\top\bot^*$ give rise to the following sets of x-modules.

$$\begin{aligned} \mathfrak{F}^{\bot}(\mathsf{Dog}) &= \{\emptyset, \{\alpha_3\}, \{\alpha_2, \alpha_3\}, \mathsf{Dog}\} \\ \mathfrak{F}^{\top}(\mathsf{Dog}) &= \{\emptyset,\ \{\alpha_1\},\ \{\alpha_1, \alpha_2\},\ \mathsf{Dog}\} \\ \mathfrak{F}^{\top\bot^*}(\mathsf{Dog}) &= \{\emptyset, \{\alpha_1\}, \{\alpha_2\}, \{\alpha_3\}, \{\alpha_1, \alpha_2\}, \{\alpha_2, \alpha_3\}, \mathsf{Dog}\} \end{aligned}$$

In each of the three corresponding x-ADs, the atoms are $\{\alpha_1\}$, $\{\alpha_2\}$, and $\{\alpha_3\}$. The dependency relations differ and are given by the Hasse diagrams in Figure 3.

In the $\bot$-AD, the principal ideal of $\{\alpha_1\}$ is $\downarrow\{\alpha_1\} = \{\alpha_1, \alpha_2, \alpha_3\}$; in the other two ADs, we have $\downarrow\{\alpha_1\} = \{\alpha_1\}$. In the $\top\bot^*$-AD, the three atoms form an antichain of size 3; the other ADs have singleton antichains only.

Figure 3: $\bot$-, $\top$-, and $\top\bot^*$-AD of the Dog ontology. Arrows represent the $\succ$ relation.

We can see that both in the $\bot$-AD and in the $\top$-AD all atoms are comparable. This is in contrast with the $\top\bot^*$-AD, which is totally disconnected since each atom is incomparable with any other atom. Please note that not every "dependency-closed" set of atoms forms a module: for example, $\{\alpha_1, \alpha_3\}$ is not a $\top\bot^*$-module (the smallest module containing both α_1 and α_3 is the whole ontology). We see that the $\bot$-AD shows the dependence of axioms related to more specific concepts (e.g., `Poodle`) on axioms related to more general concepts (e.g., `Mammal`). Dually, the $\top$-AD shows the dependence of axioms related to more general concepts on axioms related to more specific concepts. The $\top\bot^*$-AD, instead, shows none of these dependencies. To sum up, the $\bot$- and the $\top$-AD reveal a subsumption-preserving kind of dependence, whilst the $\top\bot^*$-AD shows a finer notion of independence than those revealed by the other kinds of ADs. We will further investigate this phenomenon in Subsection 4.3.2.

In general, the $\bot$- and $\top$-ADs of an ontology and its $\mathcal{E}$-connection are pairwise incomparable in terms of granularity: typically, some $\top$-AD atoms are unions of $\bot$-AD atoms, but there can be $\bot$-AD atoms that overlap with several $\top$-AD atoms. This behaviour can be observed in the following example.

Example 4.7. The $\bot$-AD of the ontology Koala2 (from Figure 1) consists of 11 atoms in 1 connected component and is displayed in Figure 4a (solid lines only). The nodes are labelled with their axioms α_i, together with an informal description of the contents—in this case the terms described by the axioms. Arrows represent the "depends on" relation $\succ$.

The $\bot$-AD yields a very fine-grained decomposition of Koala2 since most atoms consist of one axiom only. For example, it is intuitive that the atoms describing `Forest` and `Habitat` are separated: the topic of forests depends on the topic of habitats (because of α_8) but not vice versa. To additionally capture the ontology's knowledge about animals, its knowledge about the `hasHabitat` relation (α_5) has to be included.

In the case of Koala2, the $\top\bot^*$-AD happens to coincide with the $\bot$-AD, and the $\top$-AD consists of only 5 atoms in 2 connected components. It is superimposed on the $\bot$-AD in Figure 4 a (dashed lines). Except for the splitting of the bottom-most atom, the $\top$-AD is a coarsening of the $\bot$-AD, and all dependencies are reversed. This effect is caused by the property of $\top$-modules being 'closed under subconcepts', while $\bot$-modules are 'closed under superconcepts' (Cuenca Grau et al., 2008).

Figure 4b shows the $\mathcal{E}$-connection of Koala2 (from Figure 2) superimposed on the $\bot$-AD. It divides the two bottom-most atoms and is otherwise a coarsening of the $\bot$-AD that is

17. Throughout this paper, we use the term *(principal) ideal* for the downwards closed subset of a partially ordered set (generated by a single element); alternative terms are lower sets, down set, etc.

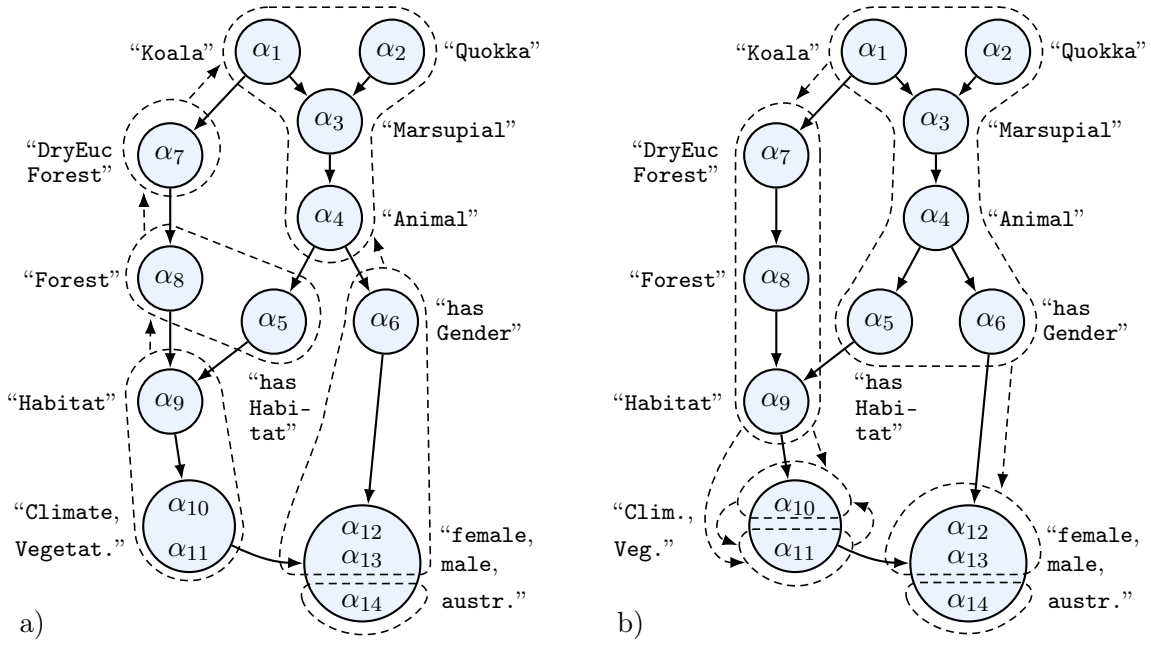


Figure 4: $\bot$-AD of the ontology Koala2 (solid lines) with a) the $\top$-AD superimposed (dashed lines) and b) the finest $\mathcal{E}$-connection superimposed. Arrows in ADs represent the $\succ$ relation.

orthogonal to the $\top$-AD. The reasons for the coarseness of the $\mathcal{E}$-connection have been discussed in Example 3.19.

Unlike the $\top$-AD and the $\mathcal{E}$-connection, the $\bot$-AD does not separate axiom α_{14} from α_{12} and α_{13}. This is an unintended consequence of the properties of $\bot$-modules: the concept assertions α_{12}–α_{14} are part of *every* $\bot$-module because concept assertions are never $\bot$-local, regardless of the signature Σ. Had the continent Australia been modelled as a concept, α_{14} would have formed a separate isolated atom because no other axiom refers to the terms `Continent` and `australia`. Modelling the genders "female" and "male" as concepts would have no such effect because axioms α_{12} and α_{13} share the concept `Gender` with each other and the remaining ontology.

Altogether, Example 4.7 shows that $\bot$-AD, $\top$-AD and decompositions based on $\mathcal{E}$-connections are pairwise incomparable. We will examine the relationship between the three main notions of ADs based on syntactic locality more systematically in Section 4.5.

The following observation is an immediate consequence of Definition 4.4: modules are downwards closed under the dependency relation $\succeq$.

Lemma 4.8. *If an atom* $\mathfrak{a} \subseteq \mathcal{M}$ *and* $\mathfrak{a} \succeq \mathfrak{b}$*, then* $\mathfrak{b} \subseteq \mathcal{M}$*.*

4.3 Properties of the Atomic Decomposition

We will discuss relevant properties of the atomic decomposition and its relation to modules, and start with a formal definition of *genuine* modules. We remind the reader that we have fixed a module notion x with the properties stated in Section 4.1. For the sake of convenience, we will omit x when referring to the set $\mathfrak{F}(\mathcal{O})$ of all modules of an ontology $\mathcal{O}$, its atom set

$\mathfrak{A}(\mathcal{O})$, or the module $\mathrm{mod}(\Sigma, \mathcal{O})$ for a signature Σ. We continue to assume that $\mathfrak{t} = \emptyset$ (i.e., every axiom occurs in some module).

Definition 4.9. Given an ontology $\mathcal{O}$, we say that a module $\mathcal{M} \in \mathfrak{F}(\mathcal{O})$ is *fake* if there are $n \geq 2$ modules $\mathcal{M}_1, \ldots, \mathcal{M}_n \in \mathfrak{F}(\mathcal{O})$ such that

- $\mathcal{M} = \mathcal{M}_1 \cup \ldots \cup \mathcal{M}_n$ and
- $\mathcal{M}_i \not\subseteq \mathcal{M}_j$ for each $i \neq j$.

A module is called *genuine* if it is not fake. The set of all genuine modules is denoted $\mathfrak{G}(\mathcal{O})$.

Definition 4.9 captures the requirement formulated in Section 4.1: a module is genuine if it is not just the union of two or more other modules that are incomparable w.r.t. set inclusion. The distinction between genuine and fake modules is legitimated by the observation that fake modules are less cohesive and thus less representative than genuine modules.

Example 4.10. For an illustration, consider the following simple Person ontology.

$$\{\ \alpha_1 : \texttt{Vegetarian} \sqsubseteq \texttt{Person}, \qquad \alpha_2 : \texttt{Student} \sqsubseteq \texttt{Person}, \qquad \alpha_3 : \texttt{Person} \sqsubseteq \texttt{Animal}\ \}$$

Every subset of Person is a module for its own signature. In particular, the singleton modules are genuine and separate the three axioms from each other, capturing the intuition that the three axioms touch on separate topics: vegetarians, students, and animals. Consequently, the concepts `Vegetarian`, `Student`, and `Animal` can be interpreted independently; if the ontology were to be extended, additional knowledge for each of the three topics could be added to the respective module. All three modules depend on the concept `Person`, which mirrors the customary situation of modules with common dependencies in software engineering. The remaining, larger modules of Person are fake and have a different status—they decompose into the smaller modules and are thus less cohesive and less representative than those.

Next we develop a 1-1 correspondence between genuine modules and atoms.

4.3.1 Atoms vs. genuine modules

We start our investigation by considering modules for seed signatures of individual axioms in an ontology: clearly, an ontology gives rise to only a linear number of such seed signatures. In the following observations and results, we consider an arbitrary ontology $\mathcal{O}$. For any axiom $\alpha \in \mathcal{O}$, let $\mathcal{M}_\alpha := \mathrm{mod}(\widetilde{\alpha}, \mathcal{O})$. We call $\mathcal{M}_\alpha$ an *axiom module*. We will first show that axiom modules play a major role among all modules, which we will then exploit in order to compute all genuine modules.

Lemma 4.11. *For every atom $\mathfrak{a} \in \mathfrak{A}(\mathcal{O})$ and every axiom $\alpha \in \mathfrak{a}$, we have that $\mathcal{M}_\alpha$ is the smallest module of $\mathcal{O}$ that contains $\mathfrak{a}$.*

Proof. We first show that $\mathfrak{a} \subseteq \mathcal{M}_\alpha$. Let $\mathfrak{a} \in \mathfrak{A}(\mathcal{O})$ and $\alpha \in \mathfrak{a}$. Since $\mathcal{O} \models \alpha$ and because of depletingness (M5), we have $\mathcal{O} \setminus \mathcal{M}_\alpha \equiv_{\widetilde{\alpha}} \emptyset$. If α is not a tautology, then $\alpha \in \mathcal{M}_\alpha$ follows directly. Otherwise, consider some module $\mathcal{M} = \mathrm{mod}(\Sigma, \mathcal{O})$ containing α (which exists because we have adopted the assumption that $\mathfrak{t} = \emptyset$). By property (M6), we also have $\alpha \in \mathrm{mod}(\Sigma \cap \widetilde{\alpha}, \mathcal{O})$, which implies $\alpha \in \mathcal{M}_\alpha$ via monotonicity (M3).

To show that $\mathcal{M}_\alpha$ is minimal with $\mathfrak{a} \subseteq \mathcal{M}_\alpha$, consider any module $\mathcal{M} = \mathrm{mod}(\Sigma, \mathcal{O})$ of $\mathcal{O}$ with $\mathfrak{a} \subseteq \mathcal{M}$. Then we have

$$\mathcal{M}_\alpha = \mathrm{mod}(\widetilde{\alpha}, \mathcal{O}) \subseteq \mathrm{mod}(\widetilde{\mathcal{M}} \cup \widetilde{\alpha}, \mathcal{O}) = \mathrm{mod}(\widetilde{\mathcal{M}}, \mathcal{O}) = \mathcal{M},$$

where the inclusion is due to monotonicity (M3), the subsequent equality follows from $\widetilde{\alpha} \subseteq \widetilde{\mathcal{M}}$ (since $\alpha \in \mathfrak{a} \subseteq \mathcal{M}$), and the last equality is due to (M4). □

Analogously to axiom modules, we can define the notion of an *atom module:* given an atom $\mathfrak{a} \in \mathfrak{A}(\mathcal{O})$, let $\mathcal{M}_\mathfrak{a} := \mathrm{mod}(\widetilde{\mathfrak{a}}, \mathcal{O})$. Interestingly, axiom and atom modules coincide, which is an immediate consequence of Lemmas 4.8 and 4.11, the definition of atoms and of $\mathcal{M}_\alpha$, and Properties (M3) and (M4):

Corollary 4.12. *For all atoms $\mathfrak{a}, \mathfrak{b} \in \mathfrak{A}(\mathcal{O})$, the following hold.*

1. *$\mathcal{M}_\mathfrak{a} = \mathcal{M}_\alpha$ for every axiom $\alpha \in \mathfrak{a}$.*
2. *$\mathfrak{a}$ depends on all the atoms contained in $\mathcal{M}_\mathfrak{a} \setminus \mathfrak{a}$.*
3. *$\mathfrak{a} \succeq \mathfrak{b}$ if and only if $\mathcal{M}_\mathfrak{a} \supseteq \mathcal{M}_\mathfrak{b}$.*
4. *For every axiom $\beta \in \mathcal{O}$, we have that $\beta \in \mathfrak{a}$ if and only if $\mathcal{M}_\beta = \mathcal{M}_\mathfrak{a}$.*
5. *For all axioms $\alpha, \beta \in \mathcal{O}$, we have that $\mathcal{M}_\alpha = \mathcal{M}_\beta$ if and only if there is an atom $\mathfrak{b}$ such that $\{\alpha, \beta\} \subseteq \mathfrak{b}$.*
6. *We have that $\mathfrak{a} = \{\alpha \mid \mathcal{M}_\alpha = \mathcal{M}_\mathfrak{a}\}$.*

Clearly, the set of all modules $\mathfrak{F}(\mathcal{O})$ of an ontology $\mathcal{O}$ determines the AD $(\mathfrak{A}(\mathcal{O}), \succ)$ of $\mathcal{O}$. We want to provide an argument that, vice versa, the AD is a representation of the (genuine) modules of $\mathcal{O}$. For the first step in that argument, we consider the principal ideal $\downarrow\mathfrak{a}$ of an atom $\mathfrak{a} \in \mathfrak{A}(\mathcal{O})$, cf. Definition 4.5.

Theorem 4.13. *For each atom $\mathfrak{a} \in \mathfrak{A}(\mathcal{O})$, $\downarrow\mathfrak{a} = \mathcal{M}_\mathfrak{a}$.*

Proof. For the "$\subseteq$" direction, observe that $\mathfrak{a} \subseteq \mathcal{M}_\alpha = \mathcal{M}_\mathfrak{a}$ for some (any) axiom $\alpha \in \mathfrak{a}$, due to Lemma 4.11 and Corollary 4.12, Point 1. Thus, by the definition of the dependency relation, we have $\downarrow\mathfrak{a} \subseteq \mathcal{M}_\mathfrak{a}$. The "$\supseteq$" direction follows from Corollary 4.12, Point 2. □

As an immediate consequence of Corollary 4.12 and Theorem 4.13, there is a 1-1 correspondence between axiom modules and atoms:

Corollary 4.14. *For every module $\mathcal{M}$ the following are equivalent.*

1. *$\mathcal{M}$ is an axiom module.*
2. *$\mathcal{M}$ is an atom module.*
3. *There is an atom $\mathfrak{a} \subseteq \mathcal{M}$ such that $\mathcal{M} = \downarrow\mathfrak{a}$.*

Given the AD of $\mathcal{O}$, we can thus easily identify each axiom module $\mathcal{M}_\alpha$ by first identifying its atom, i.e., by finding $\mathfrak{a}$ with $\alpha \in \mathfrak{a}$, and then taking $\downarrow\mathfrak{a}$. Next, we show that *each* module can be 'assembled' from atoms; we will come back to this in Section 4.4.

Lemma 4.15. *Every module $\mathcal{M}$ is the union of principal ideals of atoms from an antichain in $\mathfrak{A}(\mathcal{O})$, i.e., there is an antichain $\{\mathfrak{a}_1, \ldots, \mathfrak{a}_n\} \subseteq \mathfrak{A}(\mathcal{O})$ such that*

$$\mathcal{M} = \bigcup_{i=1}^{n} {\downarrow}\mathfrak{a}_i \,.$$

Proof. By Observation 4.3, every module $\mathcal{M}$ is a disjoint finite union of atoms $\{\mathfrak{a} \mid \mathfrak{a} \subseteq \mathcal{M}\}$. For $\{\mathfrak{a}_1, \ldots, \mathfrak{a}_k\} \subseteq \{\mathfrak{a} \mid \mathfrak{a} \subseteq \mathcal{M}\}$ the set of all maximal elements among these atoms, we clearly have that the $\mathfrak{a}_i$ form an antichain, and then $\mathcal{M} = \bigcup_{i=1}^{k} {\downarrow}\mathfrak{a}_i$ follows with Lemma 4.8. □

Lemma 4.15 says that each module $\mathcal{M}$ has a unique decomposition into incomparable principal ideals. In this sense, we can say that the set of axiom modules forms a *base* for the set $\mathfrak{F}(\mathcal{O})$ for all modules of $\mathcal{O}$. In Definition 4.9 we have already introduced the set $\mathfrak{G}(\mathcal{O})$ of genuine modules as a base for all modules of $\mathcal{O}$. Next, we see that these two notions coincide, i.e., there is a 1-1 correspondence between genuine modules and axiom modules.

Theorem 4.16. *A module is an axiom module if and only if it is a genuine module.*

Proof. Due to Corollary 4.14, it suffices to show that a module is a principal ideal if and only if it is a genuine module.

For the 'only if' direction, assume to the contrary that $\mathcal{M}$ is principal ideal ${\downarrow}\mathfrak{a}$ for some atom $\mathfrak{a}$ and that $\mathcal{M}$ is fake. Then by Definition 4.9 there are $n \geq 2$ pairwise $\subseteq$-incomparable modules $\mathcal{M}_1, \ldots, \mathcal{M}_n$ with $\mathcal{M} = \mathcal{M}_1 \cup \cdots \cup \mathcal{M}_n$ and w.l.o.g. $\mathfrak{a} \subseteq \mathcal{M}_1$. Since $\mathcal{M} = {\downarrow}\mathfrak{a}$, all atoms in $\mathcal{M}_2$ depend on $\mathfrak{a}$. Since by Observation 4.3, $\mathcal{M}_2$ is the union of certain atoms and $\mathcal{M}_1$ is closed under the dependency relation (Lemma 4.8), we obtain $\mathcal{M}_2 \subseteq \mathcal{M}_1$, which contradicts the incomparability of $\mathcal{M}_1$ and $\mathcal{M}_2$.

For the 'if' direction, we show the contrapositive. Assume that $\mathcal{M}$ is *not* a principal ideal. Due to Lemma 4.15, $\mathcal{M} = \bigcup_{i=1}^{n} {\downarrow}\mathfrak{a}_i$ for an antichain $\{\mathfrak{a}_1, \ldots \mathfrak{a}_n\} \subseteq \mathfrak{A}(\mathcal{O})$. Since $\mathcal{M}$ is not a principal ideal, we must have $n \geq 2$; since the $\mathfrak{a}_i$ form an antichain, the modules ${\downarrow}\mathfrak{a}_i$ are pairwise $\subseteq$-incomparable. Hence $\mathcal{M}$ is fake. □

The following is a summary of all our findings and an extension of Corollary 4.14.

Corollary 4.17. *There is a 1-1 correspondence between atoms and genuine modules. In particular, for every module $\mathcal{M}$ the following are equivalent.*

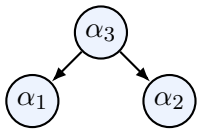
1. *$\mathcal{M}$ is a genuine module.*
2. *$\mathcal{M}$ is an axiom module.*
3. *$\mathcal{M}$ is an atom module.*
4. *There is an atom $\mathfrak{a} \subseteq \mathcal{M}$ such that $\mathcal{M} = {\downarrow}\mathfrak{a}$.*

The relations between genuine modules, atoms, and axiom modules captured in Theorem 4.16 and Corollary 4.17 will be exploited in Section 5 for the efficient computation of all genuine modules of an ontology.

We saw in Lemma 4.15 that each module is the union of the principal ideals of a suitable set of atoms. However, not all unions of principal ideals are modules:

Example 4.18. Consider the following ontology Girl and its $\bot$-AD.

$$\begin{aligned}\mathsf{Girl} = \{\, &\alpha_1 : \texttt{Female} \sqsubseteq \texttt{Person},\\ &\alpha_2 : \texttt{Child} \sqsubseteq \texttt{Person},\\ &\alpha_3 : \texttt{Female} \sqcap \texttt{Child} \sqsubseteq \texttt{Girl} \,\}\end{aligned}$$



The union of the principal ideals $\{\alpha_1\}$ and $\{\alpha_2\}$ is not a module since it includes both terms $\texttt{Female}, \texttt{Child}$ and thus needs to include the axiom α_3 too (since α_3 is not $\bot$-local w.r.t. any signature containing $\texttt{Female}, \texttt{Child}$).

4.3.2 Chains of conservative extensions

The dependency relation between (the atoms of) genuine modules has some interesting consequences for the models of these modules, which we will discuss next.

We continue to assume that we have fixed an arbitrary inseparability relation R and a corresponding module notion x with the properties (M0)–(M6) listed in Section 4.1, and we additionally require that R be monotonic (Def. 2.3). We will omit both R and x unless we need to distinguish certain choices for them explicitly.

Imagine a scenario where an ontology engineer Alice has an ontology $\mathcal{O}_1$ and an ontology $\mathcal{O}_2$ to be added to $\mathcal{O}_1$: for example, Alice has built $\mathcal{O}_1$ and now wants to extend it with $\mathcal{O}_2$ she found in a repository and which covers some important aspects of her domain. Another example is where Alice has been using $\mathcal{O}_1$ for a while and now Bob suggests to extend it with $\mathcal{O}_2$. A third example is where $\mathcal{O}_1$ is empty and Alice considers reusing $\mathcal{O}_2$ for her application.

In all three scenarios, it is reasonable to require that $\mathcal{O}_1 \cup \mathcal{O}_2$ *conservatively* extends $\mathcal{O}_1$, i.e., that $\mathcal{O}_1 \equiv^R_{\widetilde{\mathcal{O}_1}} \mathcal{O}_1 \cup \mathcal{O}_2$ for a suitable inseparability relation R. For example, if $R = \text{dCE}$, then the conservativity requirement means that all entailments of the extended ontology formulated in $\widetilde{\mathcal{O}_1}$ already follow from $\mathcal{O}_1$; hence Alice can safely restrict her attention to $\mathcal{O}_1$ when she is interested only in the 'topic' of $\mathcal{O}_1$. Similarly, if $R = \text{mCE}$, then $\{\mathcal{I} \mid \mathcal{I} \models \mathcal{O}_1\} = \{\mathcal{J}|_{\widetilde{\mathcal{O}_1}} \mid \mathcal{J} \models \mathcal{O}_1 \cup \mathcal{O}_2\}$; thus Alice can assume that her extension of $\mathcal{O}_1$ preserves the models of $\mathcal{O}_1$ modulo its signature.

Now assume that $\mathcal{O}_2$ is large and Alice wants to divide the extension into 'digestible' parts. If $\mathcal{O}_2$ contains, say, 100 axioms and Alice looks at the (unordered) set $\mathcal{O}_1 \cup \mathcal{O}_2$, then she will have a hard time to comprehend the changes and their logical effects. Instead, she may want to spread the extension over several steps: $\mathcal{O}_1 \cup \mathcal{O}_2^k$ where $\mathcal{O}_2^k$ contains the first k axioms of $\mathcal{O}_2$, then $\mathcal{O}_1 \cup \mathcal{O}_2^{2k}$ with the next k axioms, etc., up to $\mathcal{O}_1 \cup \mathcal{O}_2$. It is reasonable to expect that many small steps are preferable over few large steps; e.g., 50 steps of 2 added axioms each are easier to comprehend than 2 steps of 50 added axioms each, provided that these 50 steps of 2 axioms come *in a helpful order*. We have already seen that an order where the newly added axioms conservatively extend those already considered is such a helpful order. The following definition captures these insights.

Definition 4.19. Let R be an inseparability relation. An *R-chain of CEs* is a finite sequence of ontologies $(\mathcal{O}_1, \ldots, \mathcal{O}_m)$ such that $\mathcal{O}_i \subsetneq \mathcal{O}_{i+1}$ and $\mathcal{O}_i \equiv^R_{\widetilde{\mathcal{O}_i}} \mathcal{O}_{i+1}$ for every $i < m$.

The AD (of the ontology $\mathcal{O}$ resulting from all extensions) provides us with a straightforward mechanism for identifying chains of CEs:

Proposition 4.20. *Let $\mathcal{O}$ be an ontology and $\mathfrak{a}_1 \succ \cdots \succ \mathfrak{a}_m$ a chain of atoms. Then $(\downarrow\mathfrak{a}_m, \ldots, \downarrow\mathfrak{a}_1)$ is a chain of CEs.*

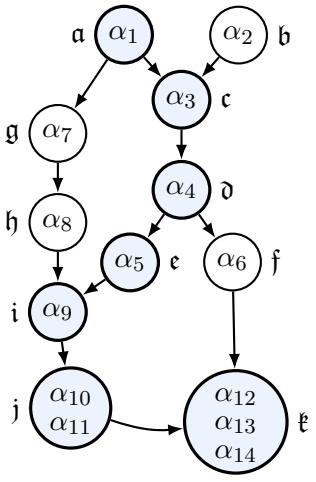
Proof. Let $i < m$. Then $\downarrow\mathfrak{a}_{i+1} \subsetneq \downarrow\mathfrak{a}_i$ is obvious from the definition of PIs and $\succ$. Furthermore, we have $\downarrow\mathfrak{a}_{i+1} \equiv_{\widetilde{\downarrow\mathfrak{a}_{i+1}}} \mathcal{O}$ due to self-containment (M0). Since $\downarrow\mathfrak{a}_{i+1} \subseteq \downarrow\mathfrak{a}_i \subseteq \mathcal{O}$ and R is monotonic, we also have $\downarrow\mathfrak{a}_{i+1} \equiv_{\widetilde{\downarrow\mathfrak{a}_{i+1}}} \downarrow\mathfrak{a}_i$. □

We can even use the AD to identify maximal chains of CEs according to Proposition 4.20 in the following sense. Consider two atoms $\mathfrak{a}_1 \prec \mathfrak{a}_2 \in \mathfrak{A}(\mathcal{O})$ such that there is no atom $\mathfrak{a}_3 \in \mathfrak{A}(\mathcal{O})$ with $\mathfrak{a}_1 \prec \mathfrak{a}_3 \prec \mathfrak{a}_2$, and let $\mathcal{M}_i = \downarrow\mathfrak{a}_i$ be the corresponding genuine modules in $\mathfrak{G}(\mathcal{O})$. Since $\mathfrak{a}_1 \prec \mathfrak{a}_2$, Proposition 4.20 ensures that $\mathcal{M}_1$ can be extended with the axioms in $\mathcal{M}_2 \setminus \mathcal{M}_1$ safely. Moreover, since there is no genuine module $\mathcal{M}_3 \in \mathfrak{G}(\mathcal{O})$ with $\mathcal{M}_1 \subsetneq \mathcal{M}_3 \subsetneq \mathcal{M}_2$, this extension is minimal (relative to the module notion x), i.e., there is no reasonable alternative way to perform this extension in two stages. This observation shows that these chains of CEs identified via the AD are of *maximal* length (and thus correspond to minimal additions), as required in the above scenario.

Example 4.21. Figure 5 a) shows again the $\bot$-AD of the Koala2 ontology (from Figures 1 and 4), with atoms numbered $\mathfrak{a}, \mathfrak{b}, \ldots, \mathfrak{k}$; the longest chain of CEs determined by the AD is of size 7 and is highlighted in the picture: $(\mathfrak{k}, \mathfrak{j}, \mathfrak{i}, \mathfrak{e}, \mathfrak{d}, \mathfrak{c}, \mathfrak{a})$. As observed above, this chain implies a suitable order for reading the axioms in Koala2, which consists of the 8 steps given in Figure 5 b).

Please note that, if we use an AD based on locality, then the corresponding inseparability relation R is model conservativity, and hence the axiom reading order induced by this AD can be used to build a model of the decomposed ontology in a strictly incremental way, i.e., without having to back-track or reconsider the interpretation of terms.

a)



b)

Step	**Axioms**	Atoms	Comments
1	$\{\alpha_{12}, \alpha_{13}, \alpha_{14}\}$	$\mathfrak{k}$	
2	$\{\alpha_{10}, \alpha_{11}\}$	$\mathfrak{j}$	
3	$\{\alpha_9\}$	$\mathfrak{i}$	
4	$\{\alpha_5\}$	$\mathfrak{e}$	
5	$\{\alpha_4, \alpha_6\}$	$\mathfrak{d}, \mathfrak{f}$	$\mathfrak{d} \succ \mathfrak{f}$
6	$\{\alpha_3\}$	$\mathfrak{c}$	
7	$\{\alpha_1, \alpha_7, \alpha_8\}$	$\mathfrak{a}, \mathfrak{g}, \mathfrak{h}$	$\mathfrak{a} \succ \mathfrak{g} \succ \mathfrak{h}$
8	$\{\alpha_2\}$	$\mathfrak{b}$	

Figure 5: a) $\bot$-AD of Koala2 (repeated from Figure 4) with longest chain of CEs; b) reading order implied by this chain.

4.3.3 Computational properties

The results described in the previous sections, in particular Theorem 4.16 (which states that there is a 1-1 correspondence between axiom modules and genuine modules) suggest that computing an AD is indeed feasible: provided that module extraction is in polynomial time (e.g., for MEX and syntactic locality based modules), it is not hard to see how we can design a polynomial algorithm for the computation of the AD. In this section, we present and discuss such an algorithm.

A naive algorithm for the computation can be designed to run in time quadratic in the number of axioms in the ontology using module extraction as an oracle by

- extracting, for each axiom, its axiom module and then
- pairwise comparing all axiom modules to determine atoms and the dependency relation.

The second step is quadratic in the number of axioms and requires a record of modules of all axioms; this is clearly infeasible for large ontologies and suboptimal for ontologies with atoms of non-trivial size.

In Algorithm 1 we give the pseudocode of an algorithm for the computation of the AD that interleaves the above two points by using so-called *key axioms* for which to record modules, and by ensuring that, for each genuine module/each atom, exactly one key axiom is identified.

The algorithm starts with an empty set $\mathsf{TauAtom}$ to record the obvious tautologies in the atom $\mathfrak{t}$, and with an empty set KeyAxs to record key axioms. It then iterates through all axioms in the ontology, extracts the module for each axiom in turn and, step by step, extends the set of key axioms and builds a record of their modules and atoms. To this end, when it considers an $\alpha \in \mathcal{O}$, it computes its axiom module $\mathsf{Module}(\alpha) = \mathcal{M}_\alpha$. If $\mathsf{Module}(\alpha) = \emptyset$[18] then α is added to $\mathsf{TauAtom}$. Otherwise, the algorithm checks whether we already have a key axiom $\beta \in \mathsf{KeyAxs}$ with the same module. If yes, α is simply added to that axiom's atom and we are done with considering α. Otherwise, we add α to the set of key axioms KeyAxs and initialise α's atom with $\{\alpha\}$. Finally, we compute the dependency relation by comparing all key axioms with all (key) axiom modules.

The correctness of this algorithm is an immediate consequence of the following observations: Corollary 4.12.4 ($\Rightarrow$) implies that creating atoms $\{\alpha\}$ for new key axioms α is correct, i.e., α does not belong into the atom of an existing key axiom. Corollary 4.12.4 ($\Leftarrow$) implies that adding axioms to the atoms of existing key axioms in case their modules coincide is correct. Then, upon termination, each axiom has been either added to $\mathsf{TauAtom}$, made a key axiom and added to its own atom, or added to the atom of a key axiom; hence we have partitioned the input ontology into correct atoms. Finally, Lemma 4.11 and Corollary 4.12.3 ensure that the dependency relation between atoms is computed correctly.

Next, we analyse the complexity of the algorithm. Let n be the number of axioms of $\mathcal{O}$. The algorithm iterates through all n axioms and extracts for each an axiom module. In the ith iteration, and in case this axiom module is non-empty, it will also compare it with at most $i-1$ modules of key axioms. Hence the 'foreach' loop F1 in Algorithm 1 is quadratic in n with n calls to the module extraction oracle (and this can be optimised further in case

18. which by Lemma 4.11 is equivalent to "α does not occur in any module"

Algorithm 1: Algorithm for computing the AD of an ontology.

input : An ontology $\mathcal{O}$; a function x-mod$(\cdot,\cdot)$ satisfying (M0)–(M6)
output: The set $\mathfrak{G}(\mathcal{O})$ of genuine modules; the poset of atoms $(\mathfrak{A}(\mathcal{O}), \succ)$; a set of key axioms KeyAxs

```
    KeyAxs ← ∅
    TauAtom ← ∅
F1  foreach α ∈ O do                          #Compute key axioms, modules & atoms
M       Module(α) ← x-mod(α̃, O)
        if Module(α) = ∅ then                 #Is α an obvious tautology?
            TauAtom ← TauAtom ∪ {α}
        else                                  #If not,
            new ← true                        #is there a key axiom β with M_α = M_β?
            foreach β ∈ KeyAxs do
                if Module(α) = Module(β) then #If yes, add α to β's atom
                    Atom(β) ← Atom(β) ∪ {α}
                    new ← false
            if new = true then                #If not, make α key
                KeyAxs ← KeyAxs ∪ {α}
                Atom(α) ← {α}                 #and start assembling its atom

    dep ← ∅                                   #Compute dependency via key axioms
F2  foreach α ∈ KeyAxs do
        foreach β ∈ KeyAxs do
            if β ∈ Module(α) then
                dep ← dep ∪ {(Atom(β), Atom(α))}

    Atoms ← {Atom(α) | α ∈ KeyAxs}            #Prepare output
    GenMods ← {Module(α) | α ∈ KeyAxs}
    return [GenMods; (Atoms, dep); KeyAxs]
```

there are several axioms with the same signature). The subsequent loop F2 is quadratic in the number of key axioms, and thus at most quadratic in n. Overall, this results in a polynomial time algorithm provided that module extraction is polynomial. The fewer atoms there are in an ontology, the fewer key axioms are considered, and hence the closer we get to a linear runtime.

So far, we have treated the module extractor as an oracle. Next, we consider a concrete example, namely the algorithm for the extraction of (non-nested) syntactic locality-based modules as described by Tsarkov in (Tsarkov, 2012), which runs in time that is linear in the *size* of an ontology. More precisely, to extract one module for a given signature, Tsarkov's module extraction considers each axiom α at most $\widetilde{\alpha}$ times, and thus runs in $\sum_{\alpha\in\mathcal{O}} |\widetilde{\alpha}|$ steps. When used for module extraction in line M of Algorithm 1, this will thus result in a runtime of $n(\sum_{\alpha\in\mathcal{O}} |\widetilde{\alpha}| + n) + n^2$. Now we can make one of two assumptions:

1. each axiom in $\mathcal{O}$ involves at most c terms, for some constant c. In this case, the runtime of Algorithm 1 is quadratic in the number of axioms in the ontology.

2. the number of terms in axioms is unbounded. In this case and for m the maximal number of terms per axiom in $\mathcal{O}$, Algorithm 1 runs in time n^2m and thus quadratic in the *size* of the ontology.

Algorithm 1 can be further optimized as described by Tsarkov in (Tsarkov, 2012): if $\mathcal{M}_\alpha$ contains an axiom $\beta \neq \alpha$ then we know that $\mathcal{M}_\beta \subseteq \mathcal{M}_\alpha$, hence we only need to check the axioms in $\mathcal{M}_\alpha$ when extracting $\mathcal{M}_\beta$. This optimization, however, does not change the asymptotic complexity.

In Section 5 we evaluate the feasibility of computing the AD in practice.

4.4 Atomic Decomposition as a modular structure

In this section, we discuss the Atomic Decomposition as a modular structure, considering the same properties as in our discussions of other modular structures in Section 3. Again, we assume we have fixed a module notion x that satisfies (M0) to (M6), for example semantic or syntactic locality (see Section 4.1), and omit x if it is clear from the context.

Modules and their properties. In this setting, genuine modules play the role of basic modules: any other x-module is a (not necessarily disjoint) union of genuine modules, see Lemma 4.15 and Corollary 4.17. Since genuine modules inherit the properties from the module notion x they are based on, they are usually not minimal in size, i.e., if one is after a module to cover all entailments over a given signature, there is usually a smaller module than one based on locality. On the other hand, they enjoy other, useful properties such as being self-contained and depleting, see the discussion in Section 2.

As observed in Section 3.3.4, the cohesion of modules depends on the choice of x: for locality-based and MEX modules, cohesion can be said to be good since axioms in the same $\mathcal{M} = x\text{-mod}(\Sigma, \mathcal{O})$ are likely to be the cause of some entailments over $\widetilde{\mathcal{M}}$, i.e., if $\mathcal{M}$ is split into two parts, there are likely to be axioms entailed by $\mathcal{M}$ that are not entailed by either of these parts.

As a consequence of Theorem 4.16, there are only linearly many genuine modules, and thus they yield a basis of possibly reasonable size–especially when compared to that described for locality based modules earlier, see Section 3.3.4.

Modular structure and its properties. In the Atomic Decomposition, the dependency relation $\succ$ on atoms provides us indeed with a dependency relation $\succ$ on (genuine) modules, see Lemma 4.8. As seen in Section 4.3.2, for ADs based on locality, $\succ$ captures model-theoretic dependencies that can be exploited for specific tasks such as determining a suitable reading order of axioms in an ontology together with an order on terms for incremental model construction.

In contrast to Δ-decompositions, basic modules of the AD can refine each other and thus we have modules at varying levels of granularity, possibly with overlap, to reflect different aspects of a given ontology.

In contrast to the decomposition based on $\mathcal{E}$-connections, an element of a model of a given ontology can be an instance of concepts from the signature of different basic modules,

e.g., we can have an instance b of `Animal`, `Marsupial`, and `Koala`, and each of the three concept names is in the (seed) signature of different atoms. This is illustrated in Figure 4.

Of course, and similar to what we have explained in Section 3.3.4, the granularity and cohesion of the AD depends on the module notion x and the modelling style of the ontology: if x generates only few, large modules from $\mathcal{O}$, then the resulting atomic decomposition will be coarse-grained and some modules will have low cohesion; if x generates smaller modules, then the granularity will be finer and cohesion higher.

The generation mechanism for ADs poses an interesting problem: given the AD $(\mathfrak{A}(\mathcal{O}), \succ)$ of $\mathcal{O}$ and a seed signature Σ, the only known mechanism to obtain $\mathrm{mod}(\Sigma, \mathcal{O})$ is to compute it from scratch, i.e., from $\mathcal{O}$; see (Del Vescovo et al., 2011; Del Vescovo, 2013) for a discussion on how labels on atoms could be used to optimise the computation of $\mathrm{mod}(\Sigma, \mathcal{O})$ in a variety of scenarios. We illustrate the difficulties via a simple example: consider $(\mathfrak{A}(\mathcal{O}), \succ)$ with three atoms $\mathfrak{a}_0 \succ \mathfrak{a}_1$ and $\mathfrak{a}_0 \succ \mathfrak{a}_2$ and $\Sigma = \widetilde{\mathfrak{a}_1} \cup \widetilde{\mathfrak{a}_2}$. The AD carries no information as to whether (i) $\mathrm{mod}(\Sigma, \mathcal{O}) = {\downarrow}\mathfrak{a}_1 \cup {\downarrow}\mathfrak{a}_2 \neq {\downarrow}\mathfrak{a}_0$ or (ii) $\mathrm{mod}(\Sigma, \mathcal{O}) = {\downarrow}\mathfrak{a}_0$. In case (i), Σ leads to a fake module which requires additional terms from $\widetilde{\mathfrak{a}_0}$ to result in a genuine module. For example, consider the $\bot$-AD and $\mathfrak{a}_1 = \{A \sqsubseteq B\}$, $\mathfrak{a}_2 = \{X \sqsubseteq Y\}$, and $\mathfrak{a}_0 = \{C \sqcap A \sqcap X \sqsubseteq \exists R.(B \sqcap Y)\}$. Then $\Sigma = \{A, B, X, Y\}$ and $\mathrm{mod}(\Sigma, \mathcal{O}) = \mathfrak{a}_1 \cup \mathfrak{a}_2$ is fake, whereas $\mathrm{mod}(\Sigma \cup \{C\}, \mathcal{O}) = {\downarrow}\mathfrak{a}_0 = \mathcal{O}$ is genuine, i.e., C is the "required additional term". In case (ii), the terms in Σ "pull in" all terms in $\widetilde{\mathfrak{a}_0}$ and thus lead to a genuine module ${\downarrow}\mathfrak{a}_0$. Take the previous example and drop "$C \sqcap$" from the axiom in $\mathfrak{a}_0$ to obtain such a case.

Computational Properties As discussed in Section 4.3.3, we can compute the atomic decomposition of an ontology in time polynomial in the size of the ontology modulo a subroutine for module extraction and, under modest assumptions w.r.t. the complexity of that subroutine, in quadratic time. In the next section, we investigate the feasibility of computing the AD for existing ontologies.

4.5 AD for particular module notions

In this subsection, we summarise observations relevant for the empirical evaluation of the atomic decomposition based on different notions of locality in the next section. We also discuss other notions of modules and how they can be used as the basis for atomic decompositions; in particular we discuss reachability- and datalog-based modules and MEX modules.

4.5.1 AD for locality-based modules

There is a strong relationship between the ADs based on the three main notions of syntactic locality, which generalises the observation made in Figure 4:

- Atoms of the $\bot$- and $\top$-AD are unions of one or more atoms of the $\top\bot^*$-AD (Proposition 4.22).
- The dependency relation of the $\top\bot^*$-AD is a refinement of the dependency relations of the $\bot$- and $\top$-ADs; it contains additional edges only between $\top\bot^*$-atoms that are part of the same $\bot$-atom, or $\top$-atom, respectively (Proposition 4.23).

In the following, we fix an ontology $\mathcal{O}$. Given an axiom $\alpha \in \mathcal{O}$, we write $\mathcal{M}^*_\alpha$, $\mathcal{M}^\bot_\alpha$, and $\mathcal{M}^\top_\alpha$ for the axiom module $\mathcal{M}_\alpha$ w.r.t. to $\top\bot^*$-, $\bot$-, or $\top$-locality, respectively.

Proposition 4.22. *For every $\top\bot^*$-atom $\mathfrak{a}$, there is a $\bot$-atom $\mathfrak{b}$ and a $\top$-atom $\mathfrak{c}$ with $\mathfrak{a} \subseteq \mathfrak{b}$ and $\mathfrak{a} \subseteq \mathfrak{c}$.*

Proof. We prove only the $\bot$-case; the $\top$-case is analogous. Let $\mathfrak{a}$ be a $\top\bot^*$-atom. From the definitions of atoms, axiom modules, and $\top\bot^*$-modules it follows that, for every $\alpha \in \mathfrak{a}$, we have $\mathfrak{a} \subseteq \mathcal{M}^*_\alpha \subseteq \mathcal{M}^\bot_\alpha$. Now consider two arbitrary axioms $\alpha, \beta \in \mathfrak{a}$. Since $\mathcal{M}^\bot_\alpha$ is the smallest $\bot$-module containing α (Lemma 4.11) and $\beta \in \mathfrak{a} \subseteq \mathcal{M}^\bot_\alpha$, every $\bot$-module that contains α contains β too. Since α, β were chosen arbitrarily, we also have that every $\bot$-module that contains β contains α too. Hence any two axioms from $\mathfrak{a}$ occur in the same $\bot$-atom, which is the desired $\mathfrak{b}$. □

Proposition 4.23. *Let $\mathfrak{a}_1, \mathfrak{a}_2$ be $\top\bot^*$-atoms with $\mathfrak{a}_1 \succ \mathfrak{a}_2$, and let $\mathfrak{b}_1, \mathfrak{b}_2$ ($\mathfrak{c}_1, \mathfrak{c}_2$) be the $\bot$-atoms ($\top$-atoms) containing $\mathfrak{a}_1$ and $\mathfrak{a}_2$. Then $\mathfrak{b}_1 = \mathfrak{b}_2$ or $\mathfrak{b}_1 \succ \mathfrak{b}_2$, and $\mathfrak{c}_1 = \mathfrak{c}_2$ or $\mathfrak{c}_1 \succ \mathfrak{c}_2$.*

Proof. Again, we prove only the $\bot$-case. Let $\alpha \in \mathfrak{a}_1$ and $\beta \in \mathfrak{a}_2$. From the definition of $\top\bot^*$-modules, we get $\mathcal{M}^*_\alpha \subseteq \mathcal{M}^\bot_\alpha$. Since $\mathfrak{a}_1 \succ \mathfrak{a}_2$, we have $\beta \in \mathcal{M}^\bot_\alpha$. Now since $\mathcal{M}^\bot_\beta$ is the smallest $\bot$-module containing β (Lemma 4.11), it follows that $\mathcal{M}^\bot_\beta \subseteq \mathcal{M}^\bot_\alpha$. Now Points 1 and 3 of Corollary 4.12 imply $\mathfrak{b}_1 = \mathfrak{b}_2$ in case $\mathcal{M}^\bot_\beta = \mathcal{M}^\bot_\alpha$ and $\mathfrak{b}_1 \succ \mathfrak{b}_2$ otherwise. □

4.5.2 AD for MEX modules

The situation for MEX modules (Konev et al., 2008) is similar to that of locality-based modules: MEX modules satisfy Properties (M0)–(M6) and are thus suitable as a base for the AD. However, we are not aware of any study of MEX-based AD.

4.5.3 AD for reachability-based modules

Reachability-based modules (RBMs) violate (M0) and (M5) in general (Nortjé et al., 2013); hence an RBM-based AD does not necessarily have the properties established in Section 4.3, which are crucial for an efficient implementation, such as the sufficiency of computing and comparing only genuine modules. But even if (M0) and (M5) were satisfied by RBMs, the RBM-based AD would still be incomparable to the AD based on the previously mentioned module notions because the hypergraph-based procedure relies on the ontology being in a specific normal form. This normal form has been developed for expressive DLs, in particular $\mathcal{SRIQ}$ and $\mathcal{SROIQ}$, together with a transformation from an arbitrary $\mathcal{SROIQ}$ ontology $\mathcal{O}$ into an ontology $\mathcal{O}'$ in normal form that is a model-conservative extension w.r.t. $\widetilde{\mathcal{O}}$ (Nortjé et al., 2013; Martín-Recuerda & Walther, 2014). To the best of our knowledge, it is unclear how the AD of $\mathcal{O}'$ relates to the AD of $\mathcal{O}$: it is obvious that the AD of $\mathcal{O}'$ may be far more fine-grained than the AD of $\mathcal{O}$, but the exact nature of this relationship remains open.

4.5.4 AD for modules based on Datalog reasoning

It is also natural to ask whether AD can be based on the suite of module notions introduced by Armas Romero et al. (2016) and described in Section 2.3—from now on called *datalog-based modules (DBMs)* for short. Although DBMs generalise locality-based modules, we are not aware of any work that has deployed DBMs for AD. Such a deployment would not be straightforward because it is not obvious which DBM variants, apart from the one corresponding to $\bot$-locality, satisfy all Properties (M0)–(M6). (M2) is trivially satisfied

and (M0) can be achieved (Armas Romero et al., 2016, Section 9). In contrast, (M5) is problematic since DBMs are *weakly* depleting, while we require what Armas Romero et al. call *strongly* depleting. Moreover, DBMs rely on a strong form of normalisation for $\mathcal{SROIQ}$. The situation differs from that of RBMs insofar as (a) the normalisation rules are different, and (b) Armas Romero et al. show that a (weakly depleting) module for a $\mathcal{SROIQ}$ ontology $\mathcal{O}$ can be recovered from one such module of the normalisation $\mathcal{O}'$ of $\mathcal{O}$; the former module is, however, no longer uniquely determined, thus no longer guaranteeting (M1).

5. Experimental evaluation

In this section, we complement the conceptual and theoretical contributions from the previous sections with an empirical study. For this purpose, we use an implementation of the decomposition algorithm described in Algorithm 1 using syntactic locality-based modules ($\bot$, $\top$, and $\top\bot^*$), run it on a suitable corpus of ontologies, and evaluate the feasibility of this algorithm as well as the structure and granularity of the resulting ADs.

5.1 Experimental setting

Corpus To build our corpus, we started with the snapshot of the NCBO BioPortal ontology repository[19] by Matentzoglu and Parsia (2017), which contains 438 ontologies. In a first step, we removed all ABox axioms (concept and role assertions) from these 438 ontologies: since our re-use and reading scenarios are aimed at the TBox part of an ontology, we want to understand how the AD behaves on TBoxes. From the resulting ontologies, we removed, in a second step, 18 empty ontologies (i.e., these were ontologies that consisted of ABox axioms only) plus a further 69 ontologies that were not in OWL 2 DL (i.e., they were strictly in OWL 2 Full for a variety of reasons). This left us with a corpus of 351 ontologies.

Table 1 shows the number of TBox axioms in the ontologies of our corpus ("size") and the length of these ontologies. The length of an ontology is the number of characters needed to write it down in DL notation, where every term name, logical operator, and quantified role counts as one character, and inclusion/equivalence symbols do not count; e.g., the axiom $A \sqsubseteq \exists R.B$ has length 3; for details see page 24 of (Del Vescovo, 2013). We show percentiles to illustrate the range of ontologies in our corpus: for any given n and measure m, the percentile Pn is the highest $m(\mathcal{O})$ among the n% ontologies $\mathcal{O}_i$ with the lowest $m(\mathcal{O}_i)$; in particular, P50 is also known as the median and P100 as the maximum.

In Figure 6, we show the OWL 2 profiles that the ontologies in our corpus fall into: while this is only an indication of their logical richness, we notice that roughly 49% of our ontologies (171) do not belong to any profile (i.e., are strictly in OWL 2 DL), and that 40% of our ontologies (69) are in all profiles, i.e., these are very close to atomic concept and role hierarchies.

Compared to the corpus used in (Del Vescovo et al., 2011), we notice that our corpus has more than twice the number of ontologies, and also many more larger ones.

Implementation We evaluate the implementation of Algorithm 1 that is part of the OWL API, namely the one available via Maven Central (maven.org) with an `artifactId` of `owlapi-tools`. We use OWL API Version 5.1.11 to compute $\bot$-, $\top$-, and $\top\bot^*$-AD.

19. `https://bioportal.bioontology.org/`

	Mean	StdDev	P50	P90	P95	P99	P100
Size	27,806	136,675	1,076	16,347	107,829	806,772	1,700,233
Length	72,425	375,019	2,558	39,325	259,576	1,695,510	4,798,793

Table 1: A summary of our corpus of 351 ontologies. The 50th (median), 75th, 90th, 99th and 100th (maximum) percentiles are shown for the size (i.e. number of axioms) and for the length (i.e., sum of length of axioms) of ontologies.

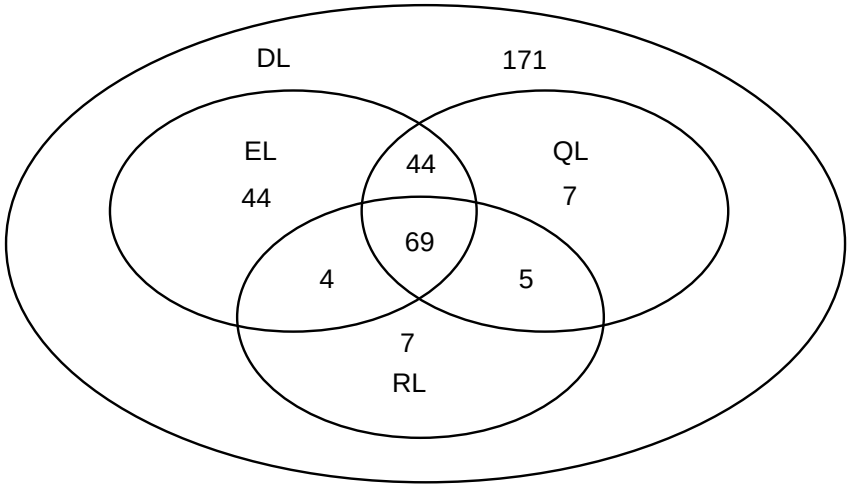


Figure 6: The spread of OWL profiles across the corpus of 351 ontologies.

All experiments have been performed on Intel(R) Core(TM) i7-6700HQ CPU 2.60GHz RAM 8GB, running Java version 1.8 JDK with an initial heap size of 1 GB and a maximal heap size of 8 GB. Time is measured in CPU time.

5.2 Feasibility

Regarding feasibility, we are interested in answering the following three questions:

1. Can we decompose all ontologies within a reasonable timeout?
2. How long does it take to decompose an ontology?
3. How does the size of an ontology affect the time needed for decomposing it?

To answer these questions, we ran the AD implementation on all ontologies in our corpus and recorded the CPU time required. The answer to the first question is 'no': 10 ontologies could not be decomposed due to a timeout. We set 6 hours as the timeout for computing each kind of AD, i.e., we used a total timeout of 18 hours for all three kinds of AD. If the ontology cannot be decomposed within 6 hours for at least one kind of AD, we say that this ontology could not be decomposed due to a timeout. In Table 2, we list these 10 ontologies, their characteristics, and the kind of AD whose computation timed out. We note that all 10 ontologies are large, i.e., have over 100k axioms, and that 6 of them are in $\mathcal{EL}$ plus possibly

Ontology	Size	Length	Profiles	Failed AD
gaz	806,772	2,410,202	EL	All
ncbitaxon	847,755	1,695,510	QL, EL, RL	$\top\bot^*$, $\bot$
cco	282,683	565,366	QL, EL, RL	$\top\bot^*$, $\top$
chebi	233,439	538,100	DL	$\top$
dron	874,945	2,655,978	DL	$\top\bot^*$, $\bot$
pxo	1,700,233	4,798,793	EL	All
reto	942,005	2,673,774	QL, EL	All
dinto	173,859	515,994	DL	$\top\bot^*$, $\bot$
ftc	145,425	461,319	EL	$\top$
biomodels	439,240	1,138,365	DL	$\top\bot^*$, $\top$

Table 2: The size (the number of TBox axioms) of ontologies that are not decomposable (all due to time-out of 18 hours), their length, profiles of those ontologies that could not be decomposed, and the decompositions that failed (e.g. for gaz, the computation of all (three) kinds of AD failed).

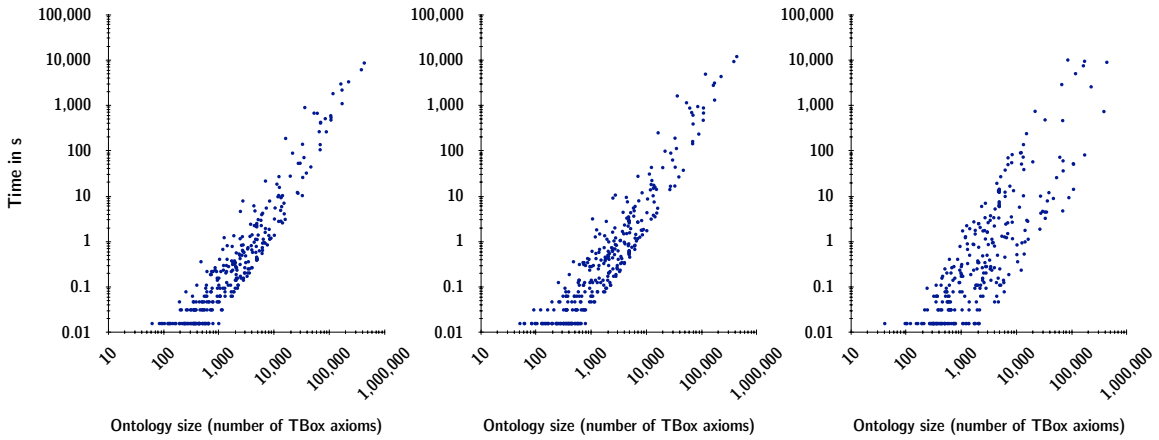
	CPU time (seconds)						
AD type	Mean	StdDev	P50	P90	P95	P99	P100
$\bot$-AD	92.51	676.76	0.11	8.67	55.66	1,919.22	9,066.98
$\top\bot^*$-AD	134.10	998.71	0.16	14.44	72.11	3,034.89	13,764.84
$\top$-AD	201.04	1,743.97	0.13	17.75	94.36	1,443.83	18,388.84

Table 3: A summary of the CPU time required for computing the ADs of the 341 decomposable ontologies in our corpus. All time are shown in seconds. Pn represents the maximum time for the nth percentile.

other profiles. Given the nature of the AD algorithm (see Algorithm 1) and module extraction algorithms, it is clear that "logical richness" should not have an effect on AD computation performance: even in QL, a single axiom can involve many terms. As we will see next, there are various ontologies that are of similar or larger size and still decomposable. Among the 10 ontologies that could not be decomposed, there are 7 ontologies that can be decomposed by at least one kind of AD computation. From now on, we will ignore all these 10 ontologies and consider only the 341 completely decomposable ontologies.

Table 3 shows the CPU time (in seconds) required for computing all three kinds of AD for the 341 decomposable ontologies from the corpus. We see that half of our ontologies take less than 1 second to decompose, 95% take under 2 minutes, and 99% take under 1 hour.

To understand how ontology size affects decomposition time, consider Figure 7: it shows scatter plots of the times for computing the three kinds of AD against ontology size, i.e., each point in each plot represents one of the 341 decomposable ontologies from our corpus, its size (x-axis) and the time it took for the computations of the AD in question (y-axis). All axes are scaled logarithmically. These plots show that the respective ADs can be computed in less than a second for almost all ontologies with up to 1,000 axioms. For the larger ontologies



(a) $\bot$-AD (b) $\top\bot^*$-AD (c) $\top$-AD

Figure 7: Plots of the runtime of AD computation against ontology size.

with up to 432,805 TBox axioms, computing the $\bot$-AD, $\top\bot^*$-AD and $\top$-AD requires up to 9,066 seconds ($\approx$2.51h), 13,764 seconds ($\approx$3.82h) and 18,388 seconds ($\approx$5.11h) respectively. The differences between the three kinds of AD are modest, but we can we can see from the shape of the scatter plots that the computation of the $\top$-AD is often faster than that of the $\bot$-AD. This can be explained by the fact that the computation of a $\top$-module is likely to be faster than that of a $\bot$-module which, in turn, is due to the following facts: the signature of $\top$-modules is 'closed under subconcepts', whereas that of $\bot$-modules is 'closed under superconcepts' (see Page 31); usually far more concepts are in the lower portion of the concept hierarchy than in the higher one; the locality-based module extraction algorithm involves an outer loop that extends the signature to terms from axioms that were found to be non-local w.r.t. the current signature, and which is thus likely to take longer to include the (longer) chain of superconcepts than the shorter chains of subconcepts. Since this effect does not occur for all ontologies, the point cloud in Figure 7 c is wider and sparser than the ones in Figure 7 a and b, but all three have the same height (for some very large ontologies the $\top$-AD requires even more time than the $\bot$-AD).

The plots in Figure 7 seem to indicate a linear growth of the computation time with respect to the ontology size, but the logarithmic scales used can be misleading. Table 4 shows trendlines and confidence intervals of the CPU time for computing the three kinds of AD. We conclude that the computation times for $\bot$-AD and $\top\bot^*$-AD grow quadratically with the size of the ontology with a high confidence ($R^2 = 96\%$ and $R^2 = 93\%$, respectively), acceptable margins (roughly 1.2 minutes and 3.1 minutes, respectively),[20] and tiny coefficients of n^2; hence we could call these two behaviours "almost linear". We also notice, however, that the confidence of linear trendlines results in a low confidence, and we checked that using cubic trendline equations do not affect them. Unsurprisingly, the confidence for the $\top$-AD is rather low and does not improve by choosing polynomials of higher degree: computation times simply vary too widely.

20. Roughly speaking, the confidence interval describes the area above and below the trendline into which other computations will fall with a probability of 95%.

AD type	Degree	R^2	Trendline equation	95% CI
$\bot$-AD	2	0.96	$3.88 \cdot 10^{-8}n^2 + 4.11 \cdot 10^{-3}n - 8.43$	$[20.68, 164.34]$
	1	0.86	$1.64 \cdot 10^{-2}n - 66.39$	
$\top\bot^*$-AD	2	0.93	$6.06 \cdot 10^{-8}n^2 + 4.49 \cdot 10^{-3}n - 4.71$	$[15.89, 386.15]$
	1	0.82	$2.36 \cdot 10^{-2}n - 95.12$	
$\top$-AD	2	0.33	$-1.34 \cdot 10^{-7}n^2 + 6.03 \cdot 10^{-2}n - 164.60$	$[28.09, 240.10]$
	1	0.16	$1.81 \cdot 10^{-2}n + 35.29$	

Table 4: A summary of the trendline equations, their determination coefficient (R^2) and the 95% confidence interval (CI) of CPU time required for computing the AD against ontology size for the 341 ontologies in our corpus.

5.3 Analysis of the structure

In this section, to understand the stucture of the different ADs, we aim at answering the following questions:

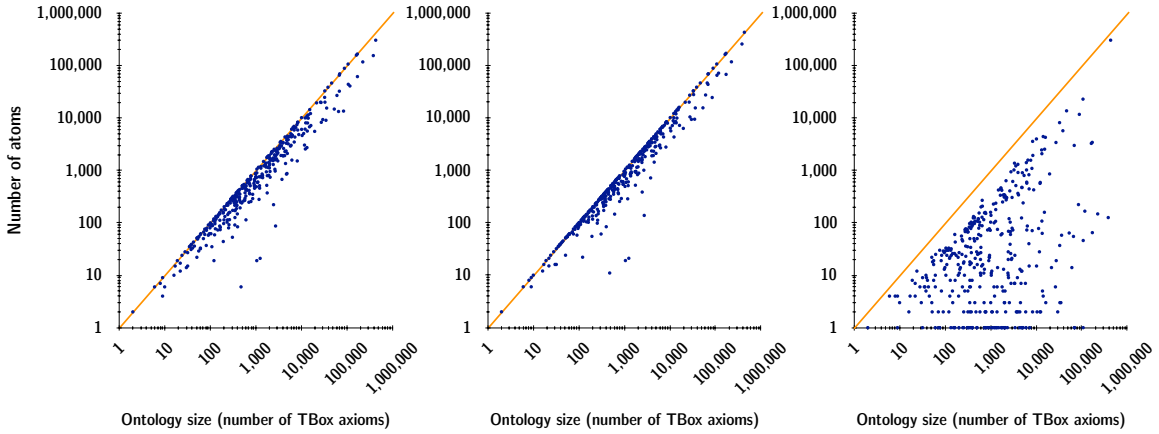
1. How many atoms do ADs have? I.e., how fine-grained is the AD?
2. How large are atoms in the AD?
3. How many incomparable (w.r.t. the dependency relation) atoms do ADs have?
4. How much structure does the AD induce? I.e., how often do we find ADs with large and/or incomparable atoms?

As in the previous section, all experiments are run on our corpus of 341 decomposable ontologies, as described in Section 5.1.

To answer Question 1, Figure 8 shows scatter plots of the number of atoms in each of the three kinds of AD against ontology size, i.e., each point in each diagram represents an ontology from the corpus and its size (x-axis) and the number of atoms in its AD (y-axis). All axes are, again, scaled logarithmically; the diagonal line marks the 'ratio 1' case, i.e., the extreme case with one atom per axiom. The $\bot$-AD (Figure 8 a) is very fine-grained, with many ontologies having ratio 1 or nearby, but there are also outliers, e.g., an ontology with 476 axioms and only 6 atoms. The $\top\bot^*$-AD (Figure 8 b) is even more fine-grained: more points lie on the 'ratio 1' line, and most other points are closer, except for the outliers. The granularity of the $\top$-AD (Figure 8 c) is quite different: it is more varied and more coarse-grained than the other two cases, i.e., there are only 3 ontologies (all of them small) with a ratio close to 1, and almost all ontologies except these three have at least twice as many axioms as they have atoms.[21] Moreover, we find 27 ontologies with a single $\top$-atom, shown by the points on the x-axis.

To answer Question 2, we consider how the size of atoms varies across the ontologies in our corpus, see Table 5. For this table, we count the number of axioms in each atom in the ADs of our ontologies, regardless of the ontology it is a part of. For $\bot$-AD and $\top\bot^*$-AD,

21. Please note that this does not imply that we have few singleton atoms as the spread of axioms across atoms may be uneven.



(a) $\bot$-AD (b) $\top\bot^*$-AD (c) $\top$-AD

Figure 8: Plots of the number of atoms against ontology size.

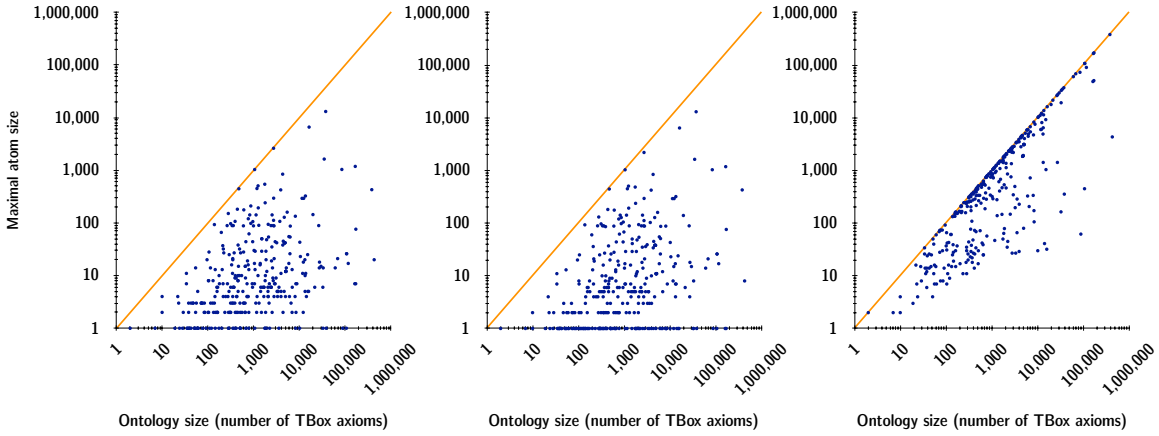
	Atom Size						
AD type	**Mean**	**StdDev**	**P50**	**P90**	**P95**	**P99**	**P100**
$\bot$-AD	1	10	1	3	4	6	13,064
$\top\bot^*$-AD	1	9	1	2	3	5	13,064
$\top$-AD	17	1,201	2	9	15	60	382,519

Table 5: A summary of the all atom size of the 341 ontologies in our corpus. Pn represents the maximum time for the nth percentile.

almost all atoms are singletons: 71.67% (for $\bot$-AD) and 87.76% (for $\top\bot^*$-AD) of atoms in this corpus have only 1 axiom. Additionally, 99% of atoms have at most 6 axioms (6 for $\bot$-AD and 5 for $\top\bot^*$-AD). For $\top$-AD, atoms tend to be larger than for $\bot$-AD and $\top\bot^*$-AD. Still, 99% atoms have no more than 60 axioms. There are, however, some very large atoms: the $\top$-AD yields an atom with 382,519 axioms.

Finally, Figure 9 shows scatter plots of the size of the maximal atom in each of the three kinds of AD against ontology size, i.e., each point in each diagram represents an ontology from our corpus and its size (x-axis) and the size of the maximal atom in its AD (y-axis). All axes are, again, scaled logarithmically. For $\bot$-AD and $\top\bot^*$-AD, the maximal atom size varies quite a bit; i.e., some ontologies have only small atoms (ontologies depicted on the x-axis have only singleton atoms), others have some medium-sized atoms, and others have a huge atom (ontologies on the diagonal have a single atom containing all axioms). As we have noticed when we considered the number of atoms in Figure 8 a and b, our corpus contains a non-trivial number of ontologies with only singleton atoms (depicted as points on the x-axis here). The $\top$-AD behaves quite differently to the other two kinds of AD since here, most ADs have a very large atom, and about a third of ontologies (29.6%) have an atom that contains almost all axioms of the ontology (at least 95%).

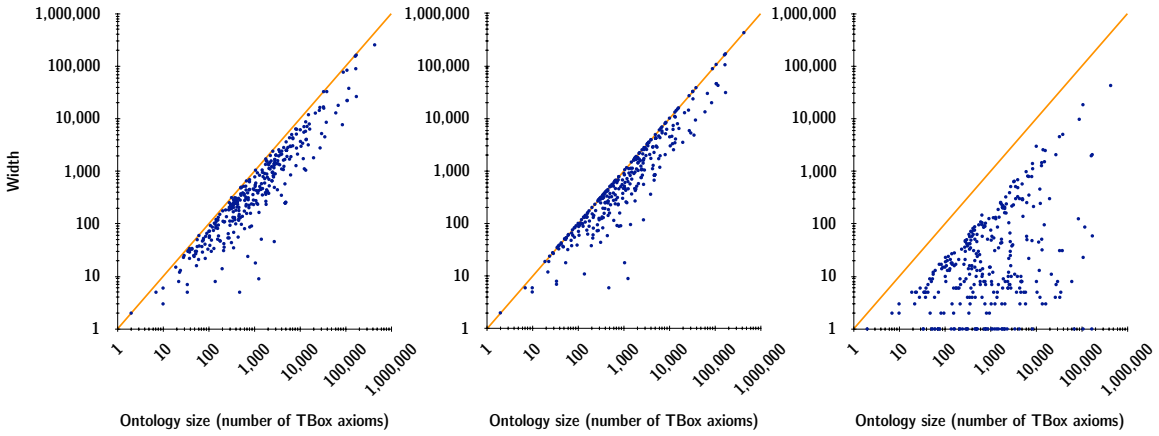
Hence the answer to Question 2 is that, for $\bot$-AD and $\top\bot^*$-AD, the size of the largest atoms varies considerably but almost all atoms are tiny, whereas for $\top$-AD, most ontologies have at least one rather large atom.



(a) $\bot$-AD (b) $\top\bot^*$-AD (c) $\top$-AD

Figure 9: Plots of maximal atom size against ontology size.

To answer Question 3, we consider antichains of atoms in $\mathfrak{A}(\mathcal{O})$, and use $\mathsf{Width}(\mathcal{O})$ to denote their maximal length. Computing maximal antichains is a known hard problem, and we use *Dilworth's theorem* (Dilworth, 2009) and the *Hopcroft–Karp algorithm* (Hopcroft & Karp, 1973) to do so.[22]



(a) $\bot$-AD (b) $\top\bot^*$-AD (c) $\top$-AD

Figure 10: Plots of AD width against ontology size.

Figure 10 shows scatter plots of the width against the size of ontologies in our corpus.[23] We find that both $\bot$-AD and $\top\bot^*$-AD often lead to very wide decompositions. In fact,

22. We implemented the computation of maximal antichains using the implementation of the Hopcroft–Karp algorithm from the Java library JGraphT.

23. Our implementation was unable to compute the width of one ontology due to an out-of-memory exception.

comparing the width with the number of atoms, we notice that, for many of our ontologies, more than half of its atoms form an antichain: this is the case for 296 out of 340 ontologies (around 87.05%) for $\bot$-AD, and 315 out of 340 (around 92.65%) for $\top\bot^*$-AD. In contrast, the width of the $\top$-AD is more evenly distributed than that of $\bot$-AD and $\top\bot^*$-AD.

To answer Question 4, we want to understand the joint effect of the individual phenomena observed above on the overall structure of the ADs, in particular how the spread of atom size and number and the length of antichains combine in the ontologies of our corpus. To this end, we consider the partial order on axioms induced by the AD, and determine how close it is to a total order. The closer the induced partial order is to a total order, the fewer large atoms the AD has, and the more atoms are related to each other via the dependency relation, i.e., the smaller the width of the AD and the more structure it has. Also, the closer the induced partial order is to a total order, the more helpful that AD will be for determining a suitable reading order, see §4.3.2.

To capture this distance, we follow the approach described in (Cook, Kress, & Seiford, 1986) to measure the distance between two orders but adapt it to our notation and simplify it (since we are not comparing two arbitrary orders). We start by computing the *comparable pairs* $\mathsf{CPs}(\mathcal{O})$ in the AD $\mathfrak{A}(\mathcal{O})$ as follows:

$$\mathsf{CPs}(\mathcal{O}) = \sum_{\mathfrak{a} \in \mathfrak{A}(\mathcal{O})} \left(\#\mathfrak{a} * \sum_{\mathfrak{b} \succ \mathfrak{a}} \#\mathfrak{b} \right). \tag{1}$$

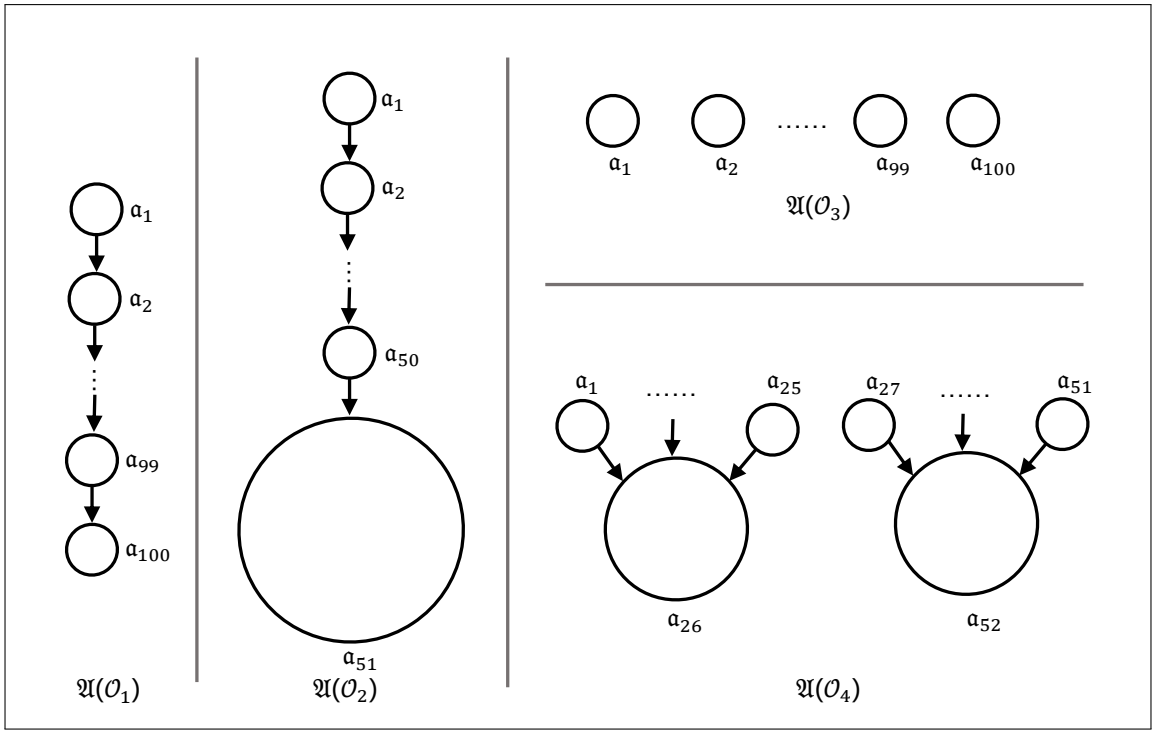


Figure 11: Four example ADs.

For example, consider the ontologies $\mathcal{O}_1, \ldots, \mathcal{O}_4$ with ADs as shown in Figure 11, and where each has 100 axioms: both $\mathfrak{A}(\mathcal{O}_1)$ and $\mathfrak{A}(\mathcal{O}_3)$ have 100 singleton atoms; in $\mathfrak{A}(\mathcal{O}_1)$ they form a chain and in $\mathfrak{A}(\mathcal{O}_3)$ they form a single antichain. In $\mathfrak{A}(\mathcal{O}_2)$, we have one long chain of 50 singleton atoms that depend on the (large) atom $\mathfrak{a}_{51}$ with 50 axioms. In $\mathfrak{A}(\mathcal{O}_4)$, we have one antichain of 50 singleton atoms which depend on the atoms $\mathfrak{a}_{26}$ and $\mathfrak{a}_{52}$ with 25 axioms each.

In $\mathfrak{A}(\mathcal{O}_1)$, $i-1$ atoms depend on $\mathfrak{a}_i$ (namely those above it), for every $i \leq 100$. Since all atoms are singletons, Equation (1) yields $\mathsf{CPs}(\mathcal{O}_1) = \sum_{i=1}^{100} \big(1 \cdot (i-1)\big) = \sum_{i=0}^{99} i = \frac{99 \cdot 100}{2} = 4,950$, i.e., all of the $\frac{99 \cdot 100}{2} = 4,950$ pairs of axioms are comparable pairs. In contrast, since each atom in $\mathfrak{A}(\mathcal{O}_3)$ depends on no other atom, there are 0 comparable pairs.

In $\mathfrak{A}(\mathcal{O}_2)$, it is again the case that $i-1$ atoms depend on $\mathfrak{a}_i$. Hence (1) yields $\mathsf{CPs}(\mathcal{O}_2) = \sum_{i=1}^{50} \big(1 \cdot (i-1)\big) + 50 \cdot (51-1) = \sum_{i=0}^{49} i + 50 \cdot 50 = 3,725$. Finally, since $\mathfrak{A}(\mathcal{O}_4)$ contains 50 singleton atoms depending on 25 axioms each (and 2 atoms depending on nothing), (1) yields $\mathsf{CPs}(\mathcal{O}_4) = 50 \cdot 25 = 1,250$.

In a next step, the *relative reading score* $\mathsf{RRS}(\mathcal{O})$ is defined as the relativised CPs defined as follows:

$$\mathsf{RRS}(\mathcal{O}) = \frac{\mathsf{CPs}(\mathcal{O})}{\big(\#\mathcal{O}^2 - \#\mathcal{O}\big)/2}.$$

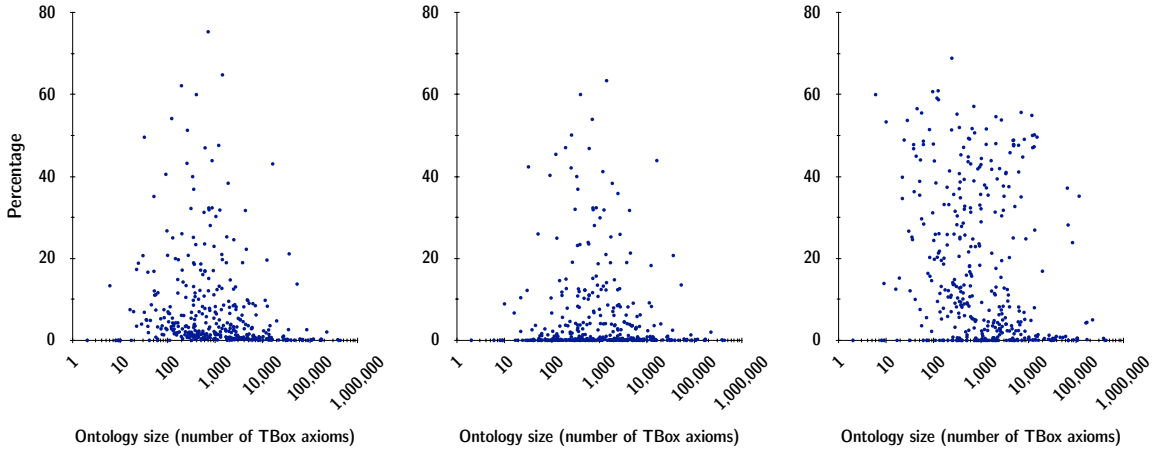
Thus, $\mathsf{RRS}(\mathcal{O})$ is the special case of the distance between the AD-induced order on axioms and a total order extending it, which has $\big(\#\mathcal{O}^2 - \#\mathcal{O}\big)/2$ comparable axiom pairs: the maximal axiom is comparable to $\#\mathcal{O} - 1$ axiom, the next smallest one to $\#\mathcal{O} - 2$, and so on, i.e. we have $(\#\mathcal{O} - 1) + (\#\mathcal{O} - 2) + \ldots + 1$ comparable pairs.

Returning to our example ontologies in Figure 11, we see that $\mathsf{RRS}(\mathcal{O}_1) = 100\%$, reflecting the fact that the AD induces a total order. Next, $\mathsf{RRS}(\mathcal{O}_3) = 0\%$ because its AD induces an empty order. Furthermore, $\mathsf{RRS}(\mathcal{O}_2) = 75.25\%$ and $\mathsf{RRS}(\mathcal{O}_4) = 25.25\%$, and hence $\mathfrak{A}(\mathcal{O}_2)$ is closer to a total order than $\mathfrak{A}(\mathcal{O}_4)$.

We have computed the RRS for all ontologies in our corpus, and Figure 12 shows scatter plots of the $\mathsf{RRS}(\mathcal{O})$ against the size of the ontology $\mathcal{O}$. We find that, for $\bot$-AD and $\top\bot^*$-AD, the RRS of almost all ontologies is below 20%, namely 298 out of 341 ontologies (around 87.39%) for $\bot$-AD, and 307 out of 341 (around 90.03%) for $\top\bot^*$-AD. For $\top$-AD, the RRS is more evenly distributed and generally higher than that of $\bot$-AD and $\top\bot^*$-AD. i.e. almost all ontologies have big atoms or/and many/large antichains w.r.t. $\bot$-AD and $\top\bot^*$-AD.

Now if we compare our RRS and Width, we find that they behave similarly for $\bot$-AD and $\top\bot^*$-AD. For example, for $\bot$-AD, 87.05% of ontologies have antichains containing at least half of the AD's atoms, and 87.39% of ontologies have a RRS below 20%. To understand better the *causes* of low reading scores, i.e., whether they are mainly caused by large atoms or long antichains, we consider the correlations between these metrics.

To do so, we need to pick a suitable *correlation coefficient*: we decide to use the well-known *Spearman's rank correlation coefficient* because we want to compare measures that grow quadratically with the ontology size like CPs with measures that are proportional to the ontology size like the width Width. In general, we would expect the correlations between the reading score or number of comparable pairs and the width/maximal atom size to be negative but modest since high values of the latter two metrics cause the former two to be low.



(a) $\bot$-AD (b) $\top\bot^*$-AD (c) $\top$-AD

Figure 12: Plots of relative reading score (RRS) against ontology size.

To compute these scores, we also introduce relativised metrics for the width, RW, and the maximal atom size, MA, so that we can compare them with the (relativised) reading score:

$$\mathsf{RW}(\mathcal{O}) = \frac{\mathsf{Width}(\mathcal{O})}{\#\mathfrak{A}(\mathcal{O})}, \qquad \mathsf{RMA}(\mathcal{O}) = \frac{\mathsf{MA}(\mathcal{O})}{\#\mathfrak{A}(\mathcal{O})}.$$

In Table 6, we use $\mathsf{Cor}(m, n)$ to represent Spearman's rank correlation coefficient between parameters m and n, and correlate the ontologies' numbers of comparable pairs CPs with their width Width (maximal length of antichains of its AD) and the size of their largest atom MA, and also correlate the corresponding two pairs of relativised metrics.

In contrast to our expections, most correlations are positive. In particular, the non-relativised metrics are all positive, for all three kinds of ADs. A reasonable explanation is that all three metrics are strongly dominated by the ontology size, i.e., the larger the ontology, the higher are CPs, Width, and MA, and thus we ignore these.

As expected, $\mathsf{Cor}(\mathsf{RRS}, \mathsf{RW})$ is negative for all three kinds of ADs and rather strong for $\top\bot^*$-AD: the latter can be explained by the fact that many ontologies have a star $\top\bot^*$-AD with only tiny atoms in a huge antichain, see Fig. 10 and 9. In contrast, this correlation is weaker for $\bot$-AD and $\top$-AD, which is due to these ADs having some large atoms (in particular $\top$-AD) and smaller width.

Similarly, for $\top$-AD, $\mathsf{Cor}(\mathsf{RRS}, \mathsf{RMA})$ is negative as expected, which can be again attributed to the prevalence of large atoms. In contrast and unexpectedly, this correlation is positive and relatively high for $\bot$-AD and $\top\bot^*$-AD, which reflects the fact that these two kinds of ADs lead to mostly tiny atoms and low RRS.

We summarize all findings from this section in Table 7. In general, we see that many ontologies in our corpus contain many unrelated atomic subsumptions which causes their $\bot$- and $\top\bot^*$-AD to contain many tiny atoms in large antichains. We also notice that the AD computation is mostly feasible, with the $\top\bot^*$-AD having the largest number of time outs and the $\top$-AD showing the largest performance variation. In general, only few of the ontologies in our corpus show a non-trivial structure – by coincidence, the two best-structured

	Correlations			
AD type	Cor(CPs, Width)	Cor(RRS, RW)	Cor(CPs, MA)	Cor(RRS, RMA)
$\bot$-AD	0.6912	-0.4555	0.6922	0.7722
$\top\bot^*$-AD	0.3447	-0.8223	0.8599	0.6534
$\top$-AD	0.6521	-0.3133	0.5945	-0.4012

Table 6: Spearman's rank correlation coefficients for CPs and Width, RRS and RW, CPs and MA, RRS and RMA, respectively.

AD type	**Feasibility**	**Number of atoms**	**Size of atoms**	**Width of AD**	**Reading Score**
$\bot$-AD	fastest, mainly quadratic	large	many tiny, some varied	many wide	mostly low
$\top\bot^*$-AD	most time-outs, mainly quadratic	large	many tiny, some varied	mostly very wide	mostly low
$\top$-AD	slowest, largest variations	varied	varied, some huge	varied, many narrow	varied

Table 7: Summary of the empirical analysis of the ADs of ontologies in our corpus.

ontologies (i.e., with the highest relative reading score RRS for all kinds of ADs) are among the oldest, most established ontologies in BioPortal: both Drosophila Development Ontology and C. elegans Development Vocabulary have been maintained since 2008 and reach RRS values of 76% and 65%, respectively, for $\bot$-AD. In general, however, the $\top$-AD is, among the three kinds of ADs, the most likely one to lead to a better structure as measured by the relative reading score.

6. Conclusion and outlook

We have introduced the atomic decomposition of an ontology and analysed its theoretical foundations, in particular the properties of the underlying module notion it relies on. We have discussed the core, general properties of a "good" modular structure of ontologies, for both the AD and related approaches. In particular, we have shown that the AD is computable in time polynomial in the size of the ontology, while its atoms provide a suitable basis for all modules, and its dependency relation has the potential to support various ontology engineering tasks, e.g., comprehension and module extraction. We have furthermore shown that the majority of ontologies from BioPortal decompose well, i.e., we can indeed compute their AD in a reasonable time, and the resulting AD is of a reasonable form.

Regarding future work, and as an immediate consequence of our feasibility analysis, it is worthwhile to improve the implementation of AD. A straightforward yet promising optimisation is one aimed at ontologies that are mostly taxonomies, i.e., ontologies that mostly consist of atomic subsumptions. Since the AD of a pure taxonomy coincides with its (inferred) concept hierarchy, we should be able to further optimise the decomposition

of "mostly taxonomic" ontologies by exploiting this relationship during parsing and thus avoiding most calls to the module extractor.

Other future work relates to applications and applicability of the AD in ontology engineering. For this purpose, we have identified two general goals, which we describe in the following.

Goal 1 is the improvement of our understanding of the AD, which can be achieved in several ways. First, the current axiom-based labels of atoms (see, e.g., Figure 4) are not very informative for tasks such as comprehension and fast module extraction. The generation of labels for each of these tasks would improve the applicability of AD. Second, our granularity analysis showed that the number of atoms can easily become too large for manual inspection. It would be useful to devise heuristics that can be employed to coarsen the decomposition, thus providing a way to control the granularity (i.e., "zoom in and out"). Third, if an ontology is being developed and its AD is to be maintained, then it would be practical to update the AD incrementally, rather than recompute it afresh after every change. Our first steps in this direction are restricted to axiom additions (Klinov et al., 2012). Fourth, we would like to "open" the AD to module notions other than locality-based modules. The family of datalog-based modules (DBMs), which generalises locality, is a natural candidate, see also Section 4.5.4. Before the AD can be based on DBMs, however, it is necessary to identify members of the DBM family that satisfy all properties (M0)–(M6). Once they are found, the AD based on them can be compared with the AD based on locality-based modules. Finally, the precise relationship of AD with related notions, such as forgetting (Zhao & Schmidt, 2018a), should be studied.

Goal 2 is the design and evaluation of methodologies that make use of AD for several tasks such as collaborative ontology development, ontology maintenance—including maintaining a suitable AD and refactoring support for decomposing an OWL ontology into a suitable set of `.owl` files (Del Vescovo et al., 2011), ontology comprehension, the identification of a suitable module for reuse/sharing, and automated reasoning and classification optimisation (Zhao, Parsia, & Sattler, 2019).

Acknowledgments

We thank Robin Nolte, Sascha Jongebloed, and the anonymous reviewers for their constructive comments on the manuscript.

References

Abiteboul, S., Hull, R., & Vianu, V. (1995). *Foundations of Databases*. Addison Wesley Publ. Co., Reading, Massachussetts.

Alchóurron, C. E., Gärdenfors, P., & Makinson, D. (1985). On the logic of theory change: Partial meet contraction and revision functions. *The Journal of Symbolic Logic*, *50*, 510–530.

Armas Romero, A., Kaminski, M., Cuenca Grau, B., & Horrocks, I. (2016). Module extraction in expressive ontology languages via Datalog reasoning. *Journal of Artificial Intelligence Research*, *55*, 499–564.

Baader, F., Calvanese, D., McGuinness, D., Nardi, D., & Patel-Schneider, P. F. (Eds.). (2007). *The Description Logic Handbook: Theory, Implementation, and Applications* (2nd edition). Cambridge University Press.

Baader, F., Horrocks, I., Lutz, C., & Sattler, U. (Eds.). (2017). *An Introduction to Description Logic*. Cambridge University Press.

Baader, F., Lutz, C., Sturm, H., & Wolter, F. (2002). Fusions of description logics and abstract description systems. *Journal of Artificial Intelligence Research*, *16*, 1–58.

Bao, J., Voutsadakis, G., Slutzki, G., & Honavar, V. (2009). Package-based Description Logics. In Stuckenschmidt et al. (Stuckenschmidt et al., 2009), pp. 349–371.

Chen, J., Alghamdi, G., Schmidt, R. A., Walther, D., & Gao, Y. (2019a). Ontology extraction for large ontologies via modularity and forgetting. In *Proceedings of the 10th International Conference on Knowledge Capture (K-CAP-19)*, pp. 45–52. ACM.

Chen, J., Ludwig, M., Ma, Y., & Walther, D. (2019b). Computing minimal projection modules for $\mathcal{ELH}^r$-terminologies. In *Proceedings of the 16th European Conference on Logics in Artificial Intelligence (JELIA-2019)*, Vol. 11468 of *Lecture Notes in Computer Science*, pp. 355–370. Springer-Verlag.

Chen, J., Ludwig, M., & Walther, D. (2018). Computing minimal subsumption modules of ontologies. In *Proceedings of the 4th Global Conference on Artificial Intelligence (GCAI-2018)*, Vol. 55 of *EPiC Series in Computing*, pp. 41–53. EasyChair.

Cook, W. D., Kress, M., & Seiford, L. M. (1986). An axiomatic approach to distance on partial orderings. *RAIRO-Operations Research*, *20*(2), 115–122.

Cuenca Grau, B., Horrocks, I., Kazakov, Y., & Sattler, U. (2008). Modular reuse of ontologies: Theory and practice. *Journal of Artificial Intelligence Research*, *31*(1), 273–318.

Cuenca Grau, B., Horrocks, I., Kazakov, Y., & Sattler, U. (2009). Extracting modules from ontologies: A logic-based approach. In Stuckenschmidt et al. (Stuckenschmidt et al., 2009), pp. 159–186.

Cuenca Grau, B., Horrocks, I., Motik, B., Parsia, B., Patel-Schneider, P. F., & Sattler, U. (2008). OWL 2: The next step for OWL. *Journal of Web Semantics*, *6*(4), 309–322.

Cuenca Grau, B., Parsia, B., & Sirin, E. (2006). Combining OWL ontologies using $\mathcal{E}$-connections. *Journal of Web Semantics*, *4*(1), 40–59.

Cuenca Grau, B., Parsia, B., & Sirin, E. (2009). Ontology integration using $\mathcal{E}$-connections. In Stuckenschmidt et al. (Stuckenschmidt et al., 2009), pp. 293–320.

Cuenca Grau, B., Parsia, B., Sirin, E., & Kalyanpur, A. (2006). Modularity and web ontologies. In *Proceedings of the 10th International Conference on the Principles of Knowledge Representation and Reasoning (KR-06)*. AAAI Press/The MIT Press.

Del Vescovo, C. (2013). *The Modular Structure of an Ontology: Atomic Decomposition and Its Applications*. Ph.D. thesis, University of Manchester. Available at `http://www.cs.man.ac.uk/~delvescc/thesis.pdf`.

Del Vescovo, C., Gessler, D., Klinov, P., Parsia, B., Sattler, U., Schneider, T., & Winget, A. (2011). Decomposition and modular structure of BioPortal ontologies. In *Proceedings*

of the 10th International Semantic Web Conference (ISWC-11), Vol. 7031 of *Lecture Notes in Computer Science*, pp. 130–145.

Del Vescovo, C., Klinov, P., Parsia, B., Sattler, U., Schneider, T., & Tsarkov, D. (2013). Empirical study of logic-based modules: Cheap is cheerful. In *Proceedings of the 12th International Semantic Web Conference (ISWC-13)*.

Del Vescovo, C., Parsia, B., Sattler, U., & Schneider, T. (2011). The modular structure of an ontology: Atomic decomposition. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI-11)*, pp. 2232–2237.

Dilworth, R. P. (2009). A decomposition theorem for partially ordered sets. In *Classic Papers in Combinatorics*, pp. 139–144. Springer.

Eiter, T., Ianni, G., Schindlauer, R., Tompits, H., & Wang, K. (2006). Forgetting in managing rules and ontologies. In *Proceedings of the 5th IEEE/WIC/ACM International Conference on Web Intelligence (WI-06)*, pp. 411–419. IEEE Computer Society.

Gatens, W., Konev, B., & Wolter, F. (2014). Lower and upper approximations for depleting modules of description logic ontologies. In *Proceedings of the 27th International Workshop on Description Logics (DL-14)*, Vol. 1193 of *CEUR (`http://ceur-ws.org/`)*.

Ghilardi, S., Lutz, C., & Wolter, F. (2006). Did I damage my ontology? A case for conservative extensions in description logics. In *Proceedings of the 10th International Conference on the Principles of Knowledge Representation and Reasoning (KR-06)*, pp. 187–197. AAAI Press/The MIT Press.

Golbeck, J., Fragoso, G., Hartel, F., Hendler, J., Oberthaler, J., & Parsia, B. (2003). The National Cancer Institute's thesaurus and ontology. *Journal of Web Semantics*, *1*(1), 75–80.

Hopcroft, J. E., & Karp, R. M. (1973). An $n^{\hat{5}}/2$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on computing*, *2*(4), 225–231.

Horrocks, I., Kutz, O., & Sattler, U. (2006). The even more irresistible $\mathcal{SROIQ}$. In *Proceedings of the 10th International Conference on the Principles of Knowledge Representation and Reasoning (KR-06)*, pp. 57–67.

Horrocks, I., Patel-Schneider, P. F., & van Harmelen, F. (2003). From SHIQ and RDF to OWL: The making of a web ontology language. *Journal of Web Semantics*, *1*(1), 7–26.

Jiménez-Ruiz, E., Cuenca Grau, B., Sattler, U., Schneider, T., & Berlanga Llavori, R. (2008). Safe and economic re-use of ontologies: A logic-based methodology and tool support. In *Proceedings of the 5th European Semantic Web Conference (ESWC-08)*, Vol. 5021 of *Lecture Notes in Computer Science*, pp. 185–199.

Jongebloed, S., & Schneider, T. (2018). Ontology partitioning using $\mathcal{E}$-Connections revisited. In *DL-18*, Vol. 2211. CEUR (`http://ceur-ws.org/`).

Klinov, P., Del Vescovo, C., & Schneider, T. (2012). Incrementally updateable and persistent decomposition of OWL ontologies. In *Proceedings of the 9th International Workshop on OWL: Experiences and Directions (OWLED-12)*, Vol. 849 of *CEUR-WS.org*.

Konev, B., Lutz, C., Ponomaryov, D., & Wolter, F. (2010). Decomposing description logic ontologies. In *Proceedings of the 12th International Conference on the Principles of Knowledge Representation and Reasoning (KR-10)*, pp. 236–246. AAAI Press/The MIT Press.

Konev, B., Lutz, C., Walther, D., & Wolter, F. (2008). Semantic modularity and module extraction in description logics. In *Proceedings of the 18th European Conference on Artificial Intelligence (ECAI-08)*, pp. 55–59.

Konev, B., Lutz, C., Walther, D., & Wolter, F. (2009). Formal properties of modularization. In Stuckenschmidt et al. (Stuckenschmidt et al., 2009), pp. 25–66.

Konev, B., Lutz, C., Walther, D., & Wolter, F. (2013). Model-theoretic inseparability and modularity of description logic ontologies. *Artificial Intelligence Journal*, *203*, 66–103.

Konev, B., Walther, D., & Wolter, F. (2009). Forgetting and uniform interpolation in large-scale description logic terminologies. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI-09)*, pp. 830–835.

Kontchakov, R., Wolter, F., & Zakharyaschev, M. (2010). Logic-based ontology comparison and module extraction, with an application to DL-Lite. *Artificial Intelligence*, *174*(15), 1093–1141.

Kontchakov, R., Pulina, L., Sattler, U., Schneider, T., Selmer, P., Wolter, F., & Zakharyaschev, M. (2009). Minimal module extraction from DL-Lite ontologies using QBF solvers. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI-09)*, pp. 836–841.

Koopmann, P., & Schmidt, R. A. (2014). Count and forget: Uniform interpolation of $\mathcal{SHQ}$-ontologies. In *Proceedings of the 7th International Joint Conference on Automated Reasoning (IJCAR-14)*, Vol. 8562 of *Lecture Notes in Computer Science*, pp. 434–448. Springer-Verlag.

Koopmann, P., & Schmidt, R. A. (2015). Uniform interpolation and forgetting for $\mathcal{ALC}$ ontologies with ABoxes. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI-15)*, pp. 175–181. AAAI Press.

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

Kutz, O., Wolter, F., & Zakharyaschev, M. (2002). Connecting abstract description systems. In *Proceedings of the 8th International Conference on the Principles of Knowledge Representation and Reasoning (KR-02)*, pp. 215–226.

Lang, J., Liberatore, P., & Marquis, P. (2003). Propositional independence: Formula-variable independence and forgetting. *Journal of Artificial Intelligence Research*, *18*, 391–443.

Lutz, C., Seylan, İ., & Wolter, F. (2012). An automata-theoretic approach to uniform interpolation and approximation in the description logic $\mathcal{EL}$. In *Proceedings of the 13th International Conference on the Principles of Knowledge Representation and Reasoning (KR-12)*. AAAI Press.

Lutz, C., Walther, D., & Wolter, F. (2007). Conservative extensions in expressive description logics. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI-07)*, pp. 453–458.

Lutz, C., & Wolter, F. (2010). Deciding inseparability and conservative extensions in the description logic $\mathcal{EL}$. *Journal of Symbolic Computation*, *45*(2), 194–228.

Lutz, C., & Wolter, F. (2011). Foundations for uniform interpolation and forgetting in expressive description logics. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI-11)*, pp. 989–995.

Martín-Recuerda, F., & Walther, D. (2014). Fast modularisation and atomic decomposition of ontologies using axiom dependency hypergraphs. In *Proceedings of the 13th International Semantic Web Conference (ISWC-14), Part II*, Vol. 8797 of *Lecture Notes in Computer Science*, pp. 49–64. Springer-Verlag.

Matentzoglu, N., & Parsia, B. (2017). BioPortal Snapshot 30 March 2017 (data set).. `http://doi.org/10.5281/zenodo.439510`.

Nikitina, N., & Rudolph, S. (2014). (Non-)succinctness of uniform interpolants of general terminologies in the description logic $\mathcal{EL}$. *Artificial Intelligence*, *215*, 120–140.

Nortjé, R., Britz, K., & Meyer, T. (2013). Reachability modules for the description logic $\mathcal{SRIQ}$. In *Proceedings of the 19th International Conference on Logic for Programming and Automated Reasoning (LPAR-19)*, Vol. 8312 of *Lecture Notes in Computer Science*, pp. 636–652. Springer-Verlag.

Parikh, R. (1999). Beliefs, belief revision, and splitting languages. In Moss, L. S., Ginzburg, J., & de Rijke, M. (Eds.), *Logic, Language and Computation*, Vol. 2, pp. 266–278. CSLI Publication, Cambridge University Press.

Parnas, D. L. (1972). On the criteria to be used in decomposing systems into modules. *Communications of the ACM*, *15*(12), 1053–1058.

Parsia, B., & Schneider, T. (2010). The modular structure of an ontology: an empirical study. In *Proceedings of the 12th International Conference on the Principles of Knowledge Representation and Reasoning (KR-10)*, pp. 584–586. AAAI Press/The MIT Press.

Poli, R., Healy, M., & Kameas, A. (Eds.). (2010). *Theory and Applications of Ontology: Computer Applications*. Springer-Verlag.

Ponomaryov, D. (2008). On decomposability in logical calculi. *Bulletin of the Novosibirsk Computing Center*, *28*, 111–120.

Reiter, R., & Lin, F. (1994). Forget it!. In *Proceedings of the AAAI Fall Symposium on Relevance*, pp. 154–159.

Sattler, U., Schneider, T., & Zakharyaschev, M. (2009). Which kind of module should I extract?. In *Proceedings of the 22nd International Workshop on Description Logics (DL-09)*, Vol. 477 of *CEUR (`http://ceur-ws.org/`)*.

Serafini, L., & Tamilin, A. (2009). Composing modular ontologies with Distributed Description Logics. In Stuckenschmidt et al. (Stuckenschmidt et al., 2009), pp. 321–347.

Spackman, K. A., Campbell, K. E., & Côté, R. A. (1997). SNOMED RT: a reference terminology for health care. In *Proceedings of the 1st American Medical Informatics Association Annual Symposium (AMIA-97)*.

Stuckenschmidt, H., Parent, C., & Spaccapietra, S. (Eds.). (2009). *Modular Ontologies: Concepts, Theories and Techniques for Knowledge Modularization*, Vol. 5445 of *Lecture Notes in Computer Science*. Springer-Verlag.

Suntisrivaraporn, B. (2008). Module extraction and incremental classification: A pragmatic approach for $\mathcal{EL}^+$ ontologies. In Bechhofer, S., Hauswirth, M., Hoffmann, J., & Koubarakis, M. (Eds.), *Proceedings of the 5th European Semantic Web Conference (ESWC-08)*, Vol. 5021 of *Lecture Notes in Computer Science*, pp. 230–244. Springer-Verlag.

ten Cate, B., Conradie, W., Marx, M., & Venema, Y. (2006). Definitorially complete description logics. In *Proceedings of the 10th International Conference on the Principles of Knowledge Representation and Reasoning (KR-06)*, pp. 79–89. AAAI Press.

Tsarkov, D. (2012). Improved algorithms for module extraction and atomic decomposition. In *Proceedings of the 25th International Workshop on Description Logics (DL-12)*, Vol. 846. CEUR (`http://ceur-ws.org/`).

Wang, K., Wang, Z., Topor, R. W., Pan, J. Z., & Antoniou, G. (2009a). Concept and role forgetting in $\mathcal{ALC}$ ontologies. In *Proceedings of the 8th International Semantic Web Conference (ISWC-09)*, Vol. 5823 of *Lecture Notes in Computer Science*, pp. 666–681. Springer-Verlag.

Wang, Z., Wang, K., Topor, R. W., Pan, J. Z., & Antoniou, G. (2009b). Uniform interpolation for $\mathcal{ALC}$ revisited. In *Proceedings of the 22nd Australasian Joint Conference on Artificial Intelligence*, Vol. 5866 of *Lecture Notes in Computer Science*, pp. 528–537. Springer-Verlag.

Zhao, H., Parsia, B., & Sattler, U. (2019). Avoiding subsumption tests during classification using the atomic decomposition. In *Proceedings of the 30th International Workshop on Description Logics (DL-19)*, Vol. 573. to appear.

Zhao, Y., & Schmidt, R. A. (2017). Role forgetting for $\mathcal{ALCOQH}(\nabla)$-ontologies using an Ackermann-based approach. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI-17)*, pp. 1354–1361. ijcai.org.

Zhao, Y., & Schmidt, R. A. (2018a). FAME: an automated tool for semantic forgetting in expressive description logics. In *Automated Reasoning - 9th International Joint Conference, IJCAR*, Vol. 10900 of *Lecture Notes in Computer Science*, pp. 19–27. Springer-Verlag.

Zhao, Y., & Schmidt, R. A. (2018b). On concept forgetting in description logics with qualified number restrictions. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI-18)*, pp. 1984–1990. ijcai.org.