Semantic Modularity and Module Extraction in Description Logics

Boris Konev[1] and **Carsten Lutz**[2] and **Dirk Walther**[1] and **Frank Wolter**[1]

Abstract. The aim of this paper is to study semantic notions of modularity in description logic (DL) terminologies and reasoning problems that are relevant for modularity. We define two notions of a module whose independence is formalised in a model-theoretic way. Focusing mainly on the DLs $\mathcal{EL}$ and $\mathcal{ALC}$, we then develop algorithms for module extraction, for checking whether a part of a terminology is a module, and for a number of related problems. We also analyse the complexity of these problems, which ranges from tractable to undecidable. Finally, we provide an experimental evaluation of our module extraction algorithms based on the large-scale terminology SNOMED CT.

1 Introduction

The main use of ontologies in computer science is to formalise the vocabulary of an application domain, i.e., to fix the vocabulary as a logical signature and to provide a logical theory that defines the meaning of terms built from the vocabulary and their relationships. To emphasise this usage, we speak of terminologies rather than of ontologies. Current applications lead to the development of large and comprehensive terminologies, as witnessed, e.g., by the Systematized Nomenclature of Medicine, Clinical Terms (SNOMED CT), which comprises ∼0.4 million terms and underlies the systematised medical vocabulary used in the health systems of the US, the UK, and other countries [13]. When working with terminologies of this size and complexity, often only a fragment of the defined vocabulary is of interest. For example, a terminology designer may want to reuse a part of a large terminology inside his own terminology without being forced to adopt it completely. If the terminology is deployed in an application, it is often also unwieldy to work with the whole terminology compared to working only with the part that is relevant for the application.

These observations illustrate the importance of the *module extraction problem* for terminologies, as studied, e.g., in [6, 2, 12, 4, 3]: given a relevant signature Σ and a terminology $\mathcal{T}$ that defines, among others, the terms from Σ, extract a minimal subset (*module*) $\mathcal{T}_0$ from $\mathcal{T}$ such that $\mathcal{T}_0$ can serve as a substitute for $\mathcal{T}$ w.r.t. Σ. What it means that $\mathcal{T}_0$ can serve as a substitute for $\mathcal{T}$ depends on the application at hand. In this paper, we aim at minimal modules $\mathcal{T}_0$ that induce the same *dependencies* between terms in Σ as the original terminology $\mathcal{T}$. We understand such dependencies in a model-theoretic way, identifying the dependencies between Σ-terms with the class of all Σ-reducts of models satisfying the terminology. Thus, two terminologies induce the same dependencies between terms in Σ if the classes of Σ-reducts of their models coincide. Applications for which the resulting type of module is appropriate include (a) importing, instead of the whole terminology, the module into another terminology; see also [6], (b) computing the classification of only the terms in the signature Σ, and (c) querying a database using the module instead of the whole terminology. The main advantage of our model-theoretic approach compared to entailment-based notions of dependencies [9, 11] is its robustness under changes to the language in which terminologies and queries are formulated.

The contribution of this paper is as follows. We introduce a model-theoretic notion of dependencies and explore the complexity of basic reasoning problems such as checking whether two terminologies induce the same dependencies and whether a terminology induces any dependencies at all. Considering terminologies formulated in the standard description logic (DL) $\mathcal{ALC}$, the lightweight DL $\mathcal{EL}$, and their extensions with inverse roles, we find that the complexity ranges from tractable via Π_2^p-complete and $\text{coNEXP}^{\text{NP}}$-complete to undecidable. Based on these notions of dependency, we introduce two notions of a module and develop algorithms for module extraction and checking whether a subset of a terminology is a module. The algorithms work on acyclic terminologies formulated in $\mathcal{ALCI}$ and $\mathcal{ELI}$. The module extraction algorithm for $\mathcal{ELI}$ has been implemented in a system called MEX, and we present experimental results comparing modules extracted from SNOMED CT by MEX with modules extracted using the $\bot$-local modules approach of [6].

Detailed proofs are provided in the technical report [8].

2 Preliminaries

In this paper, we consider the description logics $\mathcal{EL}$, $\mathcal{ELI}$, $\mathcal{ALC}$ and $\mathcal{ALCI}$. Let $\mathsf{N_C}$ and $\mathsf{N_R}$ be countably infinite and disjoint sets of *concept names* and *role names*, respectively. In $\mathcal{ALC}$, composite concepts are built up starting from the concept names in $\mathsf{N_C}$, and applying the concept constructors shown in the upper four rows of Figure 1. In the figure and in general, C and D denote concepts, and r denotes a role name. As usual, we use $\bot$ to abbreviate $\neg\top$, $\sqcup$, $\rightarrow$, and $\leftrightarrow$ for the usual Boolean connectives defined in terms of $\neg$ and $\sqcap$, and $\forall r.C$ for $\neg\exists r.\neg C$. $\mathcal{EL}$ is the fragment obtained from $\mathcal{ALC}$ by dropping negation. We obtain $\mathcal{ALCI}$ from $\mathcal{ALC}$ and $\mathcal{ELI}$ from $\mathcal{EL}$ by additionally allowing inverse roles inside existen-

[1] University of Liverpool, Liverpool, UK
[2] TU Dresden, Dresden, Germany

Name	Syntax	Semantics
top-concept	$\top$	$\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$
negation	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
existential restriction	$\exists r.C$	$\{d \in \Delta^{\mathcal{I}} \mid \exists d'\ (d,d') \in r^{\mathcal{I}} \wedge d' \in C^{\mathcal{I}}\}$
inverse role	r^-	$\{(e,d) \mid (d,e) \in r^{\mathcal{I}}\}$

Figure 1. Syntax and semantics.

tial restrictions, as shown in the bottom-most line of Figure 1.

The semantics of concepts is defined by means of *interpretations* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where the interpretation *domain* $\Delta^{\mathcal{I}}$ is a non-empty set, and $\cdot^{\mathcal{I}}$ is a function mapping each concept name A to a subset $A^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$ and each role name $r^{\mathcal{I}}$ to a binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The function $\cdot^{\mathcal{I}}$ is inductively expanded to composite concepts as shown in Figure 1.

A *general TBox* is a finite set of *axioms*, where an axiom can be either a *concept inclusion (CI)* $C \sqsubseteq D$ or a *concept equality (CE)* $C \equiv D$, with C and D concepts. If all concepts used in $\mathcal{T}$ belong to a description logic $\mathcal{L}$, then $\mathcal{T}$ is also called a general $\mathcal{L}$-TBox. An interpretation $\mathcal{I}$ *satisfies* a CI $C \sqsubseteq D$ (written $\mathcal{I} \models C \sqsubseteq D$) if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$; it *satisfies* a CE $C \equiv D$ (written $\mathcal{I} \models C \equiv D$) if $C^{\mathcal{I}} = D^{\mathcal{I}}$. $\mathcal{I}$ is a *model* of a general TBox $\mathcal{T}$ if it satisfies all axioms in $\mathcal{T}$. We write $\mathcal{T} \models C \sqsubseteq D$ ($\mathcal{T} \models C \equiv D$) if every model of $\mathcal{T}$ satisfies $C \sqsubseteq D$ ($C \equiv D$). A general TBox $\mathcal{T}$ is called a *terminology* if it satisfies the following conditions:

- all CEs are of the form $A \equiv C$ (*concept definition*) and all CIs are of the form $A \sqsubseteq C$ (*primitive concept definition*), where A is a concept name;
- no concept name occurs more than once on the left hand side of an axiom.

Define the relation $\prec_{\mathcal{T}} \subseteq \mathsf{N_C} \times (\mathsf{N_C} \cup \mathsf{N_R})$ by setting $A \prec_{\mathcal{T}} X$ iff there exists an axiom of the form $A \sqsubseteq C$ or $A \equiv C$ in $\mathcal{T}$ such that X occurs in C. Denote by $\prec_{\mathcal{T}}^*$ the transitive closure of $\prec_{\mathcal{T}}$ and set $\mathsf{depend}_{\mathcal{T}}(A) = \{X \mid A \prec_{\mathcal{T}}^* X\}$. Intuitively, $\mathsf{depend}_{\mathcal{T}}(A)$ consists of all symbols X which are used in the definition of A in $\mathcal{T}$. A terminology $\mathcal{T}$ is called *acyclic* if $A \notin \mathsf{depend}_{\mathcal{T}}(A)$ for any $A \in \mathsf{N_C}$. In an acyclic terminology $\mathcal{T}$, the set $\mathsf{Pr}(\mathcal{T})$ of *primitive* symbols in $\mathcal{T}$ consists of all role names and concept names that do not occur on the left hand side of an axiom of $\mathcal{T}$. The set $\mathsf{PPr}(\mathcal{T})$ of *pseudo-primitive* symbols in $\mathcal{T}$ consists of all symbols primitive in $\mathcal{T}$ and all A such that $A \sqsubseteq C \in \mathcal{T}$ for some C.

A *signature* Σ is a finite subset of $\mathsf{N_C} \cup \mathsf{N_R}$. The signature $\mathsf{sig}(C)$ ($\mathsf{sig}(\alpha)$, $\mathsf{sig}(\mathcal{T})$) of a concept C (axiom α, TBox $\mathcal{T}$) is the set of concept and role names which occur in C (α, $\mathcal{T}$, respectively). If $\mathsf{sig}(C) \subseteq \Sigma$, we call C a Σ-*concept* and similarly for axioms and TBoxes.

3 Semantic modularity

We introduce the fundamental notions underlying semantic dependencies and modules and give two definitions of a module. In the following, we say that two interpretations $\mathcal{I}$ and $\mathcal{J}$ *coincide on a signature* Σ, written $\mathcal{I}|_\Sigma = \mathcal{J}|_\Sigma$, iff $\Delta^{\mathcal{I}} = \Delta^{\mathcal{J}}$ and $X^{\mathcal{I}} = X^{\mathcal{J}}$ for all $X \in \Sigma$.

Definition 1. Let $\mathcal{T}_0$ and $\mathcal{T}_1$ be general TBoxes and Σ a signature.

- $\mathcal{T}_1$ is a *semantic Σ-consequence of* $\mathcal{T}_0$, written $\mathcal{T}_0 \models_\Sigma \mathcal{T}_1$, if for every model $\mathcal{I}_0$ of $\mathcal{T}_0$, there exists a model $\mathcal{I}_1$ of $\mathcal{T}_1$ with $\mathcal{I}_0|_\Sigma = \mathcal{I}_1|_\Sigma$;
- $\mathcal{T}_0$ and $\mathcal{T}_1$ are *semantically Σ-inseparable*, written $\mathcal{T}_0 \equiv_\Sigma \mathcal{T}_1$, if $\mathcal{T}_0 \models_\Sigma \mathcal{T}_1$ and $\mathcal{T}_1 \models_\Sigma \mathcal{T}_0$;
- $\mathcal{T}_1$ is a *model-conservative extension of* $\mathcal{T}_0$ w.r.t. Σ if $\mathcal{T}_1 \supseteq \mathcal{T}_0$ and $\mathcal{T}_0 \equiv_\Sigma \mathcal{T}_1$;
- $\mathcal{T}_0$ is a *semantic Σ-tautology* if $\mathcal{T}_0 \equiv_\Sigma \emptyset$.

Intuitively, two general TBoxes are semantically Σ-inseparable if they induce the same dependencies between Σ-concepts in a very strong sense: it can be shown that $\mathcal{T}_0 \equiv_\Sigma \mathcal{T}_1$ iff for every sentence φ of *second-order logic* which uses no symbols from $\mathsf{sig}(\mathcal{T}_0 \cup \mathcal{T}_1) \setminus \Sigma$, we have $\mathcal{T}_0 \models \varphi$ iff $\mathcal{T}_1 \models \varphi$. We give examples of typical applications of the notions introduced above.

Example 2. Semantic Σ-inseparability of TBoxes $\mathcal{T}_0$ and $\mathcal{T}_1$ implies that

(∗) $\mathcal{T}_0 \cup \mathcal{T} \models C \sqsubseteq D$ iff $\mathcal{T}_1 \cup \mathcal{T} \models C \sqsubseteq D$, for all TBoxes $\mathcal{T}$ and CIs $C \sqsubseteq D$ with $\mathcal{T}$, C, D formulated in any standard description logic and not using symbols from $\mathsf{sig}(\mathcal{T}_0 \cup \mathcal{T}_1) \setminus \Sigma$.

Assume, for example, that $\mathcal{T}$ is a terminology describing terms related to hospital administration which uses a set Σ of medical terms from $\mathcal{T}_1 =$ SNOMED CT. If the designer of $\mathcal{T}$ knows that $\mathcal{T}_1$ and another medical terminology $\mathcal{T}_0$ are semantically Σ-inseparable, then it does not make any difference whether he imports $\mathcal{T}_1$ or $\mathcal{T}_0$ into $\mathcal{T}$. If $\mathcal{T}_0$ is much smaller than $\mathcal{T}_1$, the latter might be preferable. Observe that the quantification over $\mathcal{T}$ in (∗) ensures that this property does not break when $\mathcal{T}$ evolves.

Example 3. It follows from (∗) that for any semantic Σ-tautology $\mathcal{T}_0$, the following holds: for all TBoxes $\mathcal{T}$ and CIs $C \sqsubseteq D$ such that $\mathcal{T}$, C, and D use no symbols from $\mathsf{sig}(\mathcal{T}_0 \cup \mathcal{T}_1) \setminus \Sigma$, $\mathcal{T} \models C \sqsubseteq D$ iff $\mathcal{T}_0 \cup \mathcal{T} \models C \sqsubseteq D$. Thus, one can import into $\mathcal{T}_0$ any such $\mathcal{T}$ without changing the dependencies that $\mathcal{T}$ induces between terms in Σ. If $\mathcal{T}_0$ is a terminology for hospital administration and a semantic Σ-tautology for a set Σ of medical terms defined in SNOMED CT, then one can import SNOMED CT into $\mathcal{T}_0$ without corrupting the meaning of medical terms defined in SNOMED CT. This application is discussed in detail in [6] under the name of *safety* for a signature Σ.

To illustrate the difference between entailment-based notions of inseparability as in [9, 11] and semantic Σ-inseparability consider the following example. Let

$$\Sigma = \{A, B\} \qquad \text{and} \qquad \mathcal{T}_1 = \{A \sqsubseteq \exists r.B\}.$$

Observe that, in models $\mathcal{I}$ of $\mathcal{T}_1$, $A^{\mathcal{I}} \neq \emptyset$ implies $B^{\mathcal{I}} \neq \emptyset$. Thus, $\mathcal{T}_1$ is not a semantic Σ-tautology. However, this dependency between A and B cannot be expressed in terms of a CI, and $\mathcal{T}_1$ entails the same Σ-CIs as the empty TBox in all of the DLs introduced in Section 2. Slightly more complex examples show that even property (∗) above is not equivalent to semantic Σ-inseparability. The exact relation between semantic

Σ-inseparability and entailment-based notions of conservative extensions has been investigated in detail in [5, 6, 9, 11].

Throughout this paper, we consider two kinds of modules.

Definition 4. Let $\mathcal{T}_0 \subseteq \mathcal{T}_1$ be general TBoxes and $\Sigma \supseteq \mathsf{sig}(\mathcal{T}_0)$. Then $\mathcal{T}_0$ is a

- *weak semantic* Σ*-module* of $\mathcal{T}_1$ iff $\mathcal{T}_1$ is semantically Σ-inseparable from $\mathcal{T}_0$;
- *strong semantic* Σ*-module* of $\mathcal{T}_1$ iff $\mathcal{T}_1 \setminus \mathcal{T}_0$ is a semantic Σ-tautology.

The requirement that $\mathcal{T}_0$ only contains Σ-symbols reflects the idea that modules should be self-contained: if an ontology $\mathcal{T}$ induces a dependency between symbols occurring in $\mathcal{T}_0$, then this dependency is induced by $\mathcal{T}_0$ already. Notions of a module in which this is not the case are of interest as well and are considered, e.g., in [2].

Lemma 5. *Every strong semantic module is a weak semantic module. The converse fails for acyclic $\mathcal{EL}$-terminologies.*

Proof. The first part is obvious. For the second part, let $\mathcal{T}_0 = \{A \equiv \top\}$, $\mathcal{T}_1 = \mathcal{T}_0 \cup \{B \sqsubseteq A\}$, and $\Sigma = \{A, B\}$. Then $\mathcal{T}_0$ is a weak semantic module of $\mathcal{T}_1$, but not a strong semantic module. □

Intuitively, the difference between weak and strong modules is that strong modules *additionally* require the ontology without the module to not induce any dependencies between symbols in Σ.

4 Deciding semantic Σ-consequence

It has been observed in [6, 11, 9] that semantic notions of entailment and inseparability as given in Definition 1 tend to be computationally difficult. Indeed, we can prove a strong undecidability result for deciding semantic Σ-tautologies using a reduction of the validity of a bimodal formula on a frame.

Theorem 6. *Given an acyclic $\mathcal{ALC}$-terminology $\mathcal{T}$, it is undecidable whether $\mathcal{T}$ is a semantic Σ-tautology. This even holds for acyclic $\mathcal{ALC}$-terminologies of the form $\{A \sqsubseteq C\}$ and for $\Sigma = \{A, r_1, r_2\}$.*

By definition of modules, Theorem 6 implies that it is not possible to decide, given acyclic $\mathcal{ALC}$-terminologies $\mathcal{T}_1$ and $\mathcal{T}_0 \subseteq \mathcal{T}_1$ and a signature Σ, whether $\mathcal{T}_0$ is a weak/strong semantic Σ-module in $\mathcal{T}_1$. Thus, Theorem 6 and related results explain why the notions of modularity from Definition 4 have not yet found practical applications. Instead, applications use notions of a module based on locality [6] or deductive versions of inseparability [2], or notions of a module that are not logic-based [12, 4, 3]. One aim of this paper is to challenge this approach by identifying relevant cases in which reasoning about modules as defined in Section 3 is decidable, and sometimes even tractable. A first observation is that avoiding roles in Σ improves the situation.

Theorem 7. *Let $\mathcal{L}$ be $\mathcal{ALC}$ or $\mathcal{ALCI}$. Given general $\mathcal{L}$-TBoxes $\mathcal{T}_1$ and $\mathcal{T}_0$ and a signature Σ with $\mathsf{sig}(\mathcal{T}_i) \cap \Sigma \subseteq \mathsf{N_C}$ for $i = 0, 1$, it is*

(1) $\text{coNExp}^{\text{NP}}$-complete to decide whether $\mathcal{T}_0 \equiv_\Sigma \mathcal{T}_1$; if Σ is fixed, then this problem is $\text{coNP}^{\text{NExp}}$-complete;

(2) Π_2^p-complete to decide whether $\mathcal{T}_0$ is a semantic Σ-tautology. The lower bound applies already to acyclic TBoxes.

Observe that deciding semantic Σ-tautologies under the restrictions given in Theorem 7 is easier than deciding satisfiability and subsumption in $\mathcal{ALC}$ w.r.t. acyclic TBoxes, which is PSpace-complete [10].

We remark that Theorem 7 is also of interest when analysing merged TBoxes, as it implies decidability of the following problem: given general $\mathcal{ALC}$-TBoxes $\mathcal{T}_0$ and $\mathcal{T}_1$ such that the set of symbols Σ shared by $\mathcal{T}_0$ and $\mathcal{T}_1$ contains only concept names, decide whether the union $\mathcal{T}_0 \cup \mathcal{T}_1$ is semantically Σ-inseparable from $\mathcal{T}_0$ and $\mathcal{T}_1$.

5 Deciding semantic modules

Theorem 7 suggests that controlling the role names in Σ can help to overcome undecidability of semantic modules. We identify a syntactic restriction that is inspired by this observation and recovers decidability of semantic modules for acyclic terminologies formulated in $\mathcal{ALC}$ and $\mathcal{ALCI}$. It also provides the basis for showing that, in $\mathcal{EL}$ and $\mathcal{ELI}$, we can decide semantic modules for acyclic terminologies without any further restrictions.

Definition 8. Let $\mathcal{T}$ be an acyclic terminology and $\Sigma, \Sigma_1, \Sigma_2$ signatures. $\mathcal{T}$ *contains a syntactic* (Σ_1, Σ_2)*-dependency* if there exists a concept name $A \in \Sigma_1$ such that $\mathsf{depend}_\mathcal{T}(A) \cap \Sigma_2 \neq \emptyset$. A syntactic (Σ, Σ)-dependency is called a *syntactic Σ-dependency*.

The notion of a syntactic (Σ_1, Σ_2)-dependency generalises the notion of acyclicity ($A \notin \mathsf{depend}_\mathcal{T}(A)$) to pairs of sets of symbols. Syntactic Σ-dependencies give rise to a natural case in which semantic modules in $\mathcal{ALCI}$ are decidable.

Theorem 9. *Let $\mathcal{L}$ be $\mathcal{ALC}$ or $\mathcal{ALCI}$. For acyclic $\mathcal{L}$-terminologies $\mathcal{T}_1 \supseteq \mathcal{T}_0$ and signature $\Sigma \supseteq \mathsf{sig}(\mathcal{T}_0)$ such that $\mathcal{T}_1 \setminus \mathcal{T}_0$ contains no syntactic $(\Sigma, \Sigma \cap \mathsf{N_R})$-dependencies, it is*

(1) decidable in $\text{coNExp}^{\text{NP}}$ whether $\mathcal{T}_0$ is a weak semantic Σ-module of $\mathcal{T}_1$; this problem is coNExpTime-hard;

(2) Π_2^p-complete to decide whether $\mathcal{T}_0$ is a strong semantic Σ-module of $\mathcal{T}_1$.

We conjecture that the problem in Point (1) is actually $\text{coNExp}^{\text{NP}}$-complete. It is natural to consider also a stronger syntactic condition, namely to disallow *any* Σ-dependency instead of only $(\Sigma, \Sigma \cap \mathsf{N_R})$-dependencies. In this case, the notions of strong and weak semantic modules coincide and deciding modules is only Π_2^p-complete for acyclic $\mathcal{ALC}$- and $\mathcal{ALCI}$-terminologies.

Theorem 10. *Let $\mathcal{L}$ be $\mathcal{ALC}$ or $\mathcal{ALCI}$. For acyclic $\mathcal{L}$-terminologies $\mathcal{T}_1 \supseteq \mathcal{T}_0$ and a signature $\Sigma \supseteq \mathsf{sig}(\mathcal{T}_0)$ such that $\mathcal{T}_1 \setminus \mathcal{T}_0$ contains no syntactic Σ-dependencies, the following are equivalent:*

- *$\mathcal{T}_0$ is a strong semantic Σ-module of $\mathcal{T}_1$;*
- *$\mathcal{T}_0$ is a weak semantic Σ-module of $\mathcal{T}_1$;*
- *for all $P \subseteq \Sigma \cap (\mathsf{Pr}(\mathcal{T}_0) \setminus \mathsf{Pr}(\mathcal{T}_1))$, the following concept is satisfiable in a model of $\mathcal{T}_1 \setminus \mathcal{T}_0$ of cardinality 1:*

$$C_P = \bigsqcap_{A \in P} A \sqcap \bigsqcap_{A \in \Sigma \cap (\mathsf{Pr}(\mathcal{T}_0) \setminus (\mathsf{Pr}(\mathcal{T}_1) \cup P))} \neg A.$$

It is Π_2^p-complete to decide whether $\mathcal{T}_0$ is a weak/strong semantic module of $\mathcal{T}_1$.

Output "not module" if any of the two conditions applies, and "module" otherwise:

1. there exists $A \in \Sigma \cap (\mathsf{Pr}(\mathcal{T}_0) \setminus \mathsf{Pr}(\mathcal{T}_1))$ such that $\mathsf{depend}_{\mathcal{T}_1 \setminus \mathcal{T}_0}(A) \cap \Sigma \neq \emptyset$;
2. there exists $A \in \Sigma \cap (\mathsf{Pr}(\mathcal{T}_0) \setminus \mathsf{Pr}(\mathcal{T}_1))$ such that $A \equiv C \in \mathcal{T}_1$ for some C and

$$\bigcup_{B \in \Sigma \cap (\mathsf{Pr}(\mathcal{T}_0) \setminus (\mathsf{Pr}(\mathcal{T}_1) \cup \{A\}))} \mathsf{depend}_{\mathcal{T}_1 \setminus \mathcal{T}_0}(B) \supseteq \mathsf{depend}^{\equiv}_{\mathcal{T}_1 \setminus \mathcal{T}_0}(A) \cap \mathsf{PPr}(\mathcal{T}_1 \setminus \mathcal{T}_0)$$

Figure 2. Module checking in $\mathcal{ELI}$

We now consider $\mathcal{EL}$ and $\mathcal{ELI}$. Theorems 6 and 9 show that, in the case of acyclic $\mathcal{ALCI}$-TBoxes, $(\Sigma, \Sigma \cap \mathsf{N_R})$-dependencies are the culprit for undecidability of semantic Σ-tautologies. In $\mathcal{EL}$ and $\mathcal{ELI}$, the situation is rather different. Here, dealing with Σ-dependencies (and thus also $(\Sigma, \Sigma \cap \mathsf{N_R})$-dependencies) is trivial.

Lemma 11. *Let $\mathcal{L}$ be $\mathcal{EL}$ or $\mathcal{ELI}$. If $\mathcal{T}$ is an acyclic $\mathcal{L}$-terminology that contains a syntactic Σ-dependency, then $\mathcal{T}$ is not a semantic Σ-tautology.*

Proof. In any model $\mathcal{I}$ of an acyclic $\mathcal{ELI}$-terminology $\mathcal{T}$, from $X \in \mathsf{depend}_{\mathcal{T}}(A)$ and $X^{\mathcal{I}} = \emptyset$ it follows that $A^{\mathcal{I}} = \emptyset$. The lemma follows immediately. □

Based on Lemma 11, we show that in $\mathcal{EL}$ and $\mathcal{ELI}$, modules can be decided and extracted in polytime. In what follows, we work with acyclic $\mathcal{EL}$-terminologies $\mathcal{T}$ that *contain no trivial axioms*, i.e., no axiom in $\mathcal{T}$ is of the form $A \equiv \top$ (nor $A \equiv \top \sqcap \top$, etc.). In acyclic $\mathcal{EL}$-terminologies, any such A can be eliminated by replacing it with $\top$. Thus, it is harmless to disregard trivial axioms.

Theorem 12. *Let $\mathcal{L}$ be $\mathcal{EL}$ or $\mathcal{ELI}$. For acyclic $\mathcal{L}$-terminologies $\mathcal{T}_1 \supseteq \mathcal{T}_0$ containing no trivial axioms and signature $\Sigma \supseteq \mathsf{sig}(\mathcal{T}_0)$, the following are equivalent:*

- *$\mathcal{T}_0$ is a strong semantic Σ-module of $\mathcal{T}_1$;*
- *$\mathcal{T}_0$ is a weak semantic Σ-module of $\mathcal{T}_1$.*

It is decidable in polytime whether $\mathcal{T}_0$ is a weak/strong semantic module of $\mathcal{T}_1$.

The polytime bound of Theorem 12 is established by the algorithm in Figure 2, which takes as input acyclic $\mathcal{ELI}$-terminologies $\mathcal{T}_1 \supseteq \mathcal{T}_0$ and a signature $\Sigma \supseteq \mathsf{sig}(\mathcal{T}_0)$. In the formulation of the algorithm, we use the following notation. A concept name $A \in \mathsf{N_C}$ *directly $\equiv$-depends on* $X \in \mathsf{N_C} \cup \mathsf{N_R}$, in symbols $A \prec^{\equiv}_{\mathcal{T}} X$, iff there exists $A \equiv C \in \mathcal{T}$ such that X occurs in C. Then, $\mathsf{depend}^{\equiv}_{\mathcal{T}}(A)$ denotes the set of all X such that (A, X) is in the transitive closure of $\prec^{\equiv}_{\mathcal{T}}$. The algorithm takes as input acyclic $\mathcal{ELI}$-terminologies $\mathcal{T}_1 \supseteq \mathcal{T}_0$ and a signature $\Sigma \supseteq \mathsf{sig}(\mathcal{T}_0)$.

Theorem 12 yields the interesting result that, for acyclic $\mathcal{ELI}$-terminologies, checking semantic modules is computationally simpler than subsumption, which is PSPACE-complete [7].

6 Module extraction

Based on the results given in the previous section, we devise algorithms for extracting modules from acyclic $\mathcal{ALCI}$- and $\mathcal{ELI}$-terminologies. We start with $\mathcal{ALCI}$, for which an extraction algorithm is given in Figure 3. It takes as input an acyclic $\mathcal{ALCI}$-terminology $\mathcal{T}_1$ and a signature Σ, and it outputs a module $\mathcal{T}_0$ as described by the following theorem.

Initialise: $\mathcal{T}_0 = \emptyset$.
Apply Rules 1 and 2 exhaustively, preferring Rule 1
Output $\mathcal{T}_0$.

1. if $A \in \Sigma \cup \mathsf{sig}(\mathcal{T}_0)$, $\alpha \in \mathcal{T}_1 \setminus \mathcal{T}_0$ has A on the left hand side, and $\mathsf{depend}_{\mathcal{T}_1 \setminus \mathcal{T}_0}(A) \cap (\Sigma \cup \mathsf{sig}(\mathcal{T}_0)) \neq \emptyset$, set $\mathcal{T}_0 := \mathcal{T}_0 \cup \{\alpha\}$,
2. if $\alpha \in \mathcal{T}_1 \setminus \mathcal{T}_0$ with A on the left-hand side and there is a minimal subset $Q \subseteq (\Sigma \cup \mathsf{sig}(\mathcal{T}_0)) \cap (\mathsf{Pr}(\mathcal{T}_0) \setminus \mathsf{Pr}(\mathcal{T}_1))$ such that $A \in Q$ and for some $P \subseteq Q$, the concept

$$C_{P,Q} = \bigsqcap_{A \in P} A \sqcap \bigsqcap_{A \in Q \setminus P} \neg A$$

is not satisfiable in a one-point model of $\mathcal{T}_1 \setminus \mathcal{T}_0$, then set $\mathcal{T}_0 := \mathcal{T}_0 \cup \{\alpha\}$.

Figure 3. Module extraction in $\mathcal{ALCI}$

Initialise: $\mathcal{T}_0 = \emptyset$.
Apply Rules 1 and 2 exhaustively, preferring Rule 1
Output $\mathcal{T}_0$.

1. if $A \in \Sigma \cup \mathsf{sig}(\mathcal{T}_0)$, $\alpha \in \mathcal{T}_1 \setminus \mathcal{T}_0$ has A on the left hand side, and $\mathsf{depend}_{\mathcal{T}_1 \setminus \mathcal{T}_0}(A) \cap (\Sigma \cup \mathsf{sig}(\mathcal{T}_0)) \neq \emptyset$, set $\mathcal{T}_0 := \mathcal{T}_0 \cup \{\alpha\}$.
2. if $A \in \Sigma \cup \mathsf{sig}(\mathcal{T}_0)$, $A \equiv C \in \mathcal{T}_1 \setminus \mathcal{T}_0$, and

$$\bigcup_{B \in (\Sigma \cup \mathsf{sig}(\mathcal{T}_0)) \cap (\mathsf{Pr}(\mathcal{T}_0) \setminus (\mathsf{Pr}(\mathcal{T}_1) \cup \{A\}))} \mathsf{depend}_{\mathcal{T}_1 \setminus \mathcal{T}_0}(B) \supseteq \mathsf{depend}^{\equiv}_{\mathcal{T}_1 \setminus \mathcal{T}_0}(A) \cap \mathsf{PPr}(\mathcal{T}_1 \setminus \mathcal{T}_0),$$

set $\mathcal{T}_0 := \mathcal{T}_0 \cup \{A \equiv C\}$.

Figure 4. Module extraction in $\mathcal{ELI}$

Theorem 13. *Let $\mathcal{T}_1$ be an acyclic $\mathcal{ALCI}$-terminology and Σ a signature. The output $\mathcal{T}_0$ of the algorithm in Figure 3 is the unique smallest strong (equivalently, weak) semantic $\Sigma \cup \mathsf{sig}(\mathcal{T}_0)$-module of $\mathcal{T}_1$ such that $\mathcal{T}_1 \setminus \mathcal{T}_0$ contains no syntactic Σ-dependencies.*

The condition that $\mathcal{T}_1 \setminus \mathcal{T}_0$ contains no syntactic Σ-dependencies is essential. Without it, we would have smaller modules, but cannot extract them automatically. The latter follows from Theorem 6 and the fact that, without the mentioned condition, the smallest semantic Σ-module of a terminology $\mathcal{T}_1$ is empty iff $\mathcal{T}_1$ is a semantic Σ-tautology. Also observe that the module $\mathcal{T}_0$ extracted by the algorithm is not necessarily formulated in Σ, but may contain additional symbols. This is clearly unavoidable even in simple cases, e.g. when extracting a semantic $\{A, B\}$-module from the terminology $\{A \sqsubseteq B \sqcap B'\}$. If implemented carefully, the check whether Rule 2 is applicable to a given axiom $\alpha \in \mathcal{T}_1 \setminus \mathcal{T}_0$ can be done in Σ^p_2 (and is also hard for Σ^p_2). Apart from this, the algorithm runs in polynomial time.

The algorithm for module extraction in $\mathcal{ALCI}$ first checks for syntactic Σ-dependencies and then applies (a variation of) module checking. When extracting modules from $\mathcal{ELI}$-

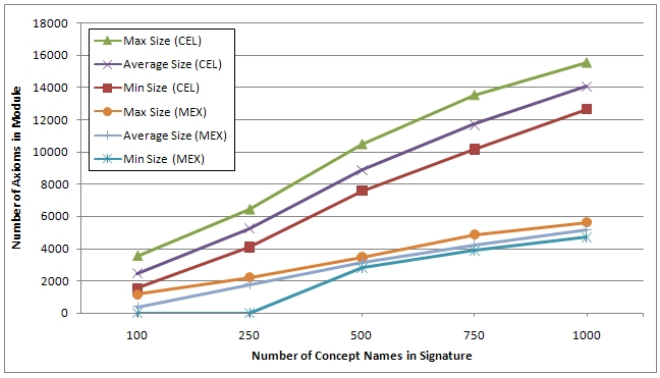

Figure 5. Sizes of $\bot$-local modules and semantic modules

terminologies, we apply the same strategy. In contrast to $\mathcal{ALCI}$, we know from Lemma 11 that if $\mathcal{T}_0 \subseteq \mathcal{T}_1$ is such that $\mathcal{T}_1 \setminus \mathcal{T}_0$ contains a Σ-dependency, then $\mathcal{T}_0$ is not a weak (equivalently, strong) Σ-module of $\mathcal{T}_1$. It follows that we do not need the additional condition on modules from Theorem 13, i.e., that $\mathcal{T}_1 \setminus \mathcal{T}_0$ contains no Σ-dependency. The extraction algorithm for $\mathcal{ELI}$ is given in Figure 4. It takes as input an acyclic $\mathcal{ELI}$-terminology $\mathcal{T}_1$ and a signature Σ, and it outputs a semantic module $\mathcal{T}_0$ as described by the following theorem.

Theorem 14. *Let $\mathcal{T}_1$ be an acyclic $\mathcal{ELI}$-terminology containing no trivial axioms and Σ a signature. The output $\mathcal{T}_0$ of the algorithm in Figure 4 is the unique smallest strong (equivalently, weak) semantic $\Sigma \cup \mathsf{sig}(\mathcal{T})$-module of $\mathcal{T}_1$.*

Example 15. Consider again the scenario described in Example 2, but now suppose that $\mathcal{T}_0$ is the output of the algorithm of Figure 4 applied to SNOMED CT and Σ. Then it does not make any difference whether the user imports $\mathcal{T}_0$ or SNOMED CT into $\mathcal{T}$. Experimental results in the next section show that $\mathcal{T}_0$ is often much smaller than SNOMED CT.

7 Experiments with MEX

To evaluate our approach to module extraction, we have carried out a number of experiments on the medical terminology SNOMED CT, an acyclic $\mathcal{EL}$-terminology that additionally comprises role inclusion statements of the form $r \sqsubseteq s$ (*role hierarchies*) and $r \circ s \sqsubseteq r$ (*right identities*). A variation of the algorithm in Figure 4 that addresses this case is presented in the technical report accompanying this paper. It was implemented in the system MEX, which is written in OCaml.

The main aim of our experiments is to compare the size of modules extracted by MEX with the size of minimal $\bot$-local modules as introduced in [6]. For $\mathcal{EL}$, $\bot$-local modules coincide with the modules extracted by the extraction feature of the CEL reasoner [14], which is used in the experiments below. We have used the SNOMED CT version of February 2005, which comprises 379 691 axioms. Experiments are based on randomly selected signatures of size between 100 and 1 000 symbols and were carried out for 1 000 different signatures of each size. Figure 5 shows the maximal, minimal, and average module sizes depending on the size of the signature. Note that, in every case, the largest semantic module is smaller than the smallest $\bot$-local module.

Additionally to generating small modules, MEX is rather efficient regarding runtime and memory consumption. We have carried out the experiments on a PC with Intel® Core™ 2 CPU at 2.13 GHz and with 3 GB of RAM. For all signature sizes in Figure 5, the average time of module extraction was 4.1 seconds and at most 124.7 MB of memory were consumed. This performance does not significantly decrease with large signature sizes: the average time and space consumed by MEX when extracting a module for 10 000 symbols in 5 seconds and 121.7 MByte. For 100 000 symbols, it is merely 9.6 seconds and 134.6 MByte.

8 Discussion

We have presented semantic notions of a module in a DL terminology and algorithms for checking and extracting such modules. Our experiments show that, at least in lightweight DLs of the $\mathcal{EL}$ family, highly efficient practical implementations of our algorithms are possible. We are optimistic that also the extraction algorithm for $\mathcal{ALCI}$ can be implemented in a reasonably efficient way.

9 Acknowledgements

The authors were supported by EPSRC grant EP/E065279/1.

REFERENCES

[1] F. Baader and C. Lutz and B. Suntisrivaraporn, 'CEL—A Polynomial-time Reasoner for Life Science Ontologies', in *Proc. of IJCAR'06*, pp. 287–291, (2006).
[2] A. Borgida, 'On importing knowdledge from DL ontologies: some intuitions and problems', in *Proc. of DL-07*, (2007).
[3] P. Doran, V. Tamma, and L.Iannone, 'Ontology module extraction for ontology reuse:an ontology engineering perspective', in *Proc. of CIKM-07*, (2007).
[4] J. Gennari et al., 'The evolution of protégé: an environment for knowledge-based systems development', *Int. J. Hum.-Comput. Stud.*, **58**(1), 89–123, (2003).
[5] B. Cuenca Grau, I. Horrocks, Y. Kazakov, and U. Sattler., 'A logical framework for modularity of ontologies', in *Proc of IJCAI'07*. AAAI Press, (2007).
[6] B. Cuenca Grau, I. Horrocks, Y. Kazakov, and U. Sattler, 'Just the right amount: extracting modules from ontologies', in *Proc. of WWW-07*, pp. 717–726, (2007).
[7] C. Haase and C. Lutz, 'Complexity of subsumption in the $\mathcal{EL}$ family of description logics: Acyclic and cyclic TBoxes', in *Proc. of ECAI-08*. (2008).
[8] B. Konev, C. Lutz, D. Walther, and F. Wolter, 'Semantic modularity and module extraction in description logics', Technical Report, (2007).
[9] C. Lutz, D. Walther, and F. Wolter, 'Conservative extensions in expressive description logics', in *Proc. of IJCAI-07*. AAAI Press, (2007).
[10] C. Lutz, 'Complexity of terminological reasoning revisited', in *Proc. of LPAR'99*, number 1705 in LNAI, pp. 181–200. Springer, (1999).
[11] C. Lutz and F. Wolter, 'Conservative extensions in the lightweight description logic $\mathcal{EL}$', in *Proc. of CADE-2007*. Springer, (2007).
[12] J. Seidenberg and A.L. Rector, 'Web ontology segmentation: analysis, classification and use', in *Proc. of WWW-06*, pp. 13–22, (2006).
[13] K.A. Spackman, 'Managing clinical terminology hierarchies using algorithmic calculation of subsumption: Experience with SNOMED-RT', *JAMIA*, (2000).
[14] B. Suntisrivaraporn, 'Module Extraction and Incremental Classification: A Pragmatic Approach for $\mathcal{EL}^+$ Ontologies', in *Proc. of ESWC-2008*. Springer, (2008).