Logical Difference and Module Extraction with CEX and MEX

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

[1] University of Liverpool, UK,
{konev, dwalther, wolter}@liverpool.ac.uk

[2] TU Dresden, Germany
lutz@tcs.inf.tu-dresden.de

1 Introduction

We present algorithms, experimental results, and an analysis of the computational complexity for the following two problems:

- The *module extraction problem*: given a terminology $\mathcal{T}_1$ and a signature Σ, extract from $\mathcal{T}_1$ a minimal self-contained terminology $\mathcal{T}_0$ such that $\mathcal{T}_1$ and $\mathcal{T}_0$ imply the same dependencies between Σ-terms.
- The *logical diff problem*: given a signature Σ and two versions $\mathcal{T}_0$ and $\mathcal{T}_1$ of a terminology, check whether $\mathcal{T}_0$ and $\mathcal{T}_1$ are logically different in the sense that they do not imply the same dependencies between Σ-terms.

There are various approaches to these problems; see, e.g., [7, 15, 3, 4] for module extraction and [14] for an algorithm comparing ontology versions. In this paper, we consider two logic-based approaches.

(1) A semantic approach according to which two terminologies imply the same dependencies between Σ-terms if the class of Σ-reducts of models of the first terminology coincides with the class of Σ-reducts of models of the second terminology. In description logic (DL), so far, all reasoning problems considered for this approach have turned out to be undecidable [12, 13]. Rather surprisingly, we show that for acyclic $\mathcal{ELI}$-terminologies deciding and extracting semantic modules is tractable. A number of extensions of this result to expressive DLs are presented as well. Using our implementation MEX and based on SNOMED CT (the Systematized Nomenclature of Medicine, Clinical Terms used in the health systems of the US, the UK, and other countries [16]), we compare the size of the extracted modules using this semantic approach with those extracted by other approaches.

(2) We consider a language-based approach according to which two terminologies imply the same dependencies between Σ-terms if they imply the same concept implications in Σ. The language-based approach distinguishes less terminologies than the semantic approach. For many DLs, the relevant reasoning problems for the language-based approach are still decidable, but harder than subsumption by at least one exponential, e.g., 2EXPTIME-complete for general $\mathcal{ALC}$ TBoxes and EXPTIME-complete for general $\mathcal{EL}$ TBoxes [12, 13]. In this paper, we show that for the language-based approach, the logical diff problem is tractable for (not necessarily acyclic) $\mathcal{EL}$-terminologies. Using our implementation CEX, we

apply this algorithm to compare distinct versions of SNOMED CT and fragments thereof. Proofs and additional results are available at [11, 10].

2 Preliminaries

We consider the DLs $\mathcal{EL}$, $\mathcal{ELI}$, $\mathcal{ALC}$, and $\mathcal{ALCI}$. For any of these, the corresponding set of concepts is constructed from countably infinite disjoint sets $\mathsf{N_C}$ of concepts names and $\mathsf{N_R}$ of role names in the usual way. A *general TBox* $\mathcal{T}$ 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 DL $\mathcal{L}$, then $\mathcal{T}$ is also called a general $\mathcal{L}$-TBox. 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 definitions) and all CIs are of the form $A \sqsubseteq C$ (primitive concept definitions), where A is a concepts 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 $A \in \mathsf{N_C}$. The set $\mathsf{Pr}(\mathcal{T})$ of *primitive* symbols in $\mathcal{T}$ consists of all role names and concept names which 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 with $A \sqsubseteq C \in \mathcal{T}$ 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 also call C a *Σ-concept* and similarly for axioms and TBoxes.

We now introduce two formalisations of the informal notion of 'dependencies between Σ-terms': a semantic one and a language-based one. Two interpretation $\mathcal{I}$ and $\mathcal{J}$ *coincide on a signature* Σ, in symbols $\mathcal{I}|_\Sigma = \mathcal{J}|_\Sigma$, if $\Delta^{\mathcal{I}} = \Delta^{\mathcal{J}}$ 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. (1) The semantic notions are as follows:*

- *$\mathcal{T}_1$ is a* semantic Σ-consequence of *$\mathcal{T}_0$, in symbols $\mathcal{T}_0 \models_\Sigma \mathcal{T}_1$, 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*, in symbols $\mathcal{T}_0 \equiv_\Sigma \mathcal{T}_1$, if $\mathcal{T}_0 \models_\Sigma \mathcal{T}_1$ and $\mathcal{T}_1 \models_\Sigma \mathcal{T}_0$;*
- *$\mathcal{T}_0$ is* a semantic Σ-tautology *if $\emptyset \equiv_\Sigma \mathcal{T}_0$.*

(2) The language-based notions are as follows: let $\mathcal{L}$ be a DL. Then

- *$\mathcal{T}_0$ Σ-*entails *$\mathcal{T}_1$ w.r.t. $\mathcal{L}$, in symbols $\mathcal{T}_0 \models_\Sigma^{\mathcal{L}} \mathcal{T}_1$, if $\mathcal{T}_1 \models C \sqsubseteq D$ implies $\mathcal{T}_0 \models C \sqsubseteq D$, for all $\mathcal{L}$-concepts C and D with $\mathsf{sig}(C, D) \subseteq \Sigma$.*

– $\mathcal{T}_0$ *and* $\mathcal{T}_1$ *are* Σ-inseparable w.r.t. $\mathcal{L}$*, in symbols* $\mathcal{T}_0 \equiv^{\mathcal{L}}_{\Sigma} \mathcal{T}_1$*, if* $\mathcal{T}_0 \models^{\mathcal{L}}_{\Sigma} \mathcal{T}_1$ *and* $\mathcal{T}_1 \models^{\mathcal{L}}_{\Sigma} \mathcal{T}_0$.
– $\mathcal{T}_0$ *is a* Σ-tautology w.r.t. $\mathcal{L}$ *if* $\mathcal{T}_0 \equiv^{\mathcal{L}}_{\Sigma} \emptyset$.

Observe that $\mathcal{T}_1$ is a *model conservative extension of* $\mathcal{T}_0$ (as defined in [12]) if $\mathcal{T}_0 \subseteq \mathcal{T}_1$ and $\mathcal{T}_0 \equiv_{\mathsf{sig}(\mathcal{T}_0)} \mathcal{T}_1$. Similarly, $\mathcal{T}_1$ is a *conservative extension* of $\mathcal{T}_0$ in $\mathcal{L}$ (as defined in [12]) if $\mathcal{T}_0 \subseteq \mathcal{T}_1$ and $\mathcal{T}_0 \equiv^{\mathcal{L}}_{\mathsf{sig}(\mathcal{T}_0)} \mathcal{T}_1$. Intuitively, $\mathcal{T}$ being a semantic Σ-tautology means that no non-trivial semantic dependencies between Σ-terms are implied by $\mathcal{T}$. It is not difficult to see that $\mathcal{T}_0 \models_{\Sigma} \mathcal{T}_1$ implies $\mathcal{T}_0 \models^{\mathcal{L}}_{\Sigma} \mathcal{T}_1$, for any standard DL $\mathcal{L}$. In fact, one can show that $\mathcal{T}_0 \models_{\Sigma} \mathcal{T}_1$ iff every second-order predicate logic sentence φ which follows from $\mathcal{T}_1$ and whose signature is contained in Σ follows from $\mathcal{T}_0$ already. While the language-based notions depend on the DL chosen and allow for the definition of rather fine-tuned notions of modularity by careful choice of the DL, the semantic notions are language independent and, therefore, the language $\mathcal{L}$ used need not be known in advance. We refer the reader to [8, 7, 6] for further information on these and related notions, and remark that the notion of Σ-tautology is called "safety for Σ" in [7].

3 Σ-entailment for $\mathcal{EL}$-terminologies

We present a polynomial-time algorithm that decides Σ-entailment for (possibly cyclic) $\mathcal{EL}$-terminologies. In this section, all concepts, concept implications, and terminologies are in $\mathcal{EL}$. Thus, we omit the prefix $\mathcal{EL}$, and '$\mathcal{T}_0$ Σ-entails $\mathcal{T}_1$' stands for '$\mathcal{T}_0$ Σ-entails $\mathcal{T}_1$ w.r.t. $\mathcal{EL}$'. An implication $C \sqsubseteq D$ is called a *counter-example to* Σ*-entailment* of $\mathcal{T}_1$ by $\mathcal{T}_0$ if C, D are Σ-concepts and $\mathcal{T}_1 \models C \sqsubseteq D$, but $\mathcal{T}_0 \not\models C \sqsubseteq D$.

Lemma 1. *Let* $\mathcal{T}_0$ *and* $\mathcal{T}_1$ *be terminologies and* Σ *a signature. If* $\mathcal{T}_0$ *does not* Σ*-entail* $\mathcal{T}_1$ *and* $C \sqsubseteq D$ *is a counter-example to* Σ*-entailment, then there exist subconcepts* C' *and* D' *of* C *and* D*, respectively, such that* $C' \sqsubseteq D'$ *is a counter-example to* Σ*-entailment of the form* $A \sqsubseteq \exists r.D_0$ *or* $C_0 \sqsubseteq A$*, where* A *is a concept name.*

It follows that the problem of deciding Σ-entailment is split into two parts: decide whether there exists a counter-example of the form $C \sqsubseteq A$, and if this is not the case, decide whether there exists a counter-example of the form $A \sqsubseteq D$. The latter problem has been shown to be decidable in polynomial-time already in [13]. It thus remains to show the following:

Theorem 1. *Given* $\mathcal{EL}$*-terminologies* $\mathcal{T}_0$ *and* $\mathcal{T}_1$*, it is decidable in polynomial-time whether there exists a counter-example to* $\mathcal{T}_0 \models^{\mathcal{EL}}_{\Sigma} \mathcal{T}_1$ *of the form* $C \sqsubseteq A$.

When proving Theorem 1, we assume that the terminology $\mathcal{T}_0$ is in a certain normal form (but this need not be the case for $\mathcal{T}_1$). A concept name A is called *non-conjunctive* in a terminology $\mathcal{T}$ if it is pseudo-primitive in $\mathcal{T}$ or has a definition of the form $A \equiv \exists r.C \in \mathcal{T}$, and *conjunctive* otherwise. We say that $\mathcal{T}$ is in *normal form* if it consists of axioms of the following form:

For each $A \in \mathsf{sig}(\mathcal{T}_0) \cup \Sigma$, define $\mathsf{noimply}_{\mathcal{T}_0,\Sigma}(A)$ and a TBox $\mathcal{T}_A$ as follows:

- If A is pseudo-primitive in $\mathcal{T}_0$, then

$$\mathsf{noimply}_{\mathcal{T}_0,\Sigma}(A) = \{\xi_A\} \quad \text{and} \quad \mathcal{T}_A = \Big\{\xi_A \sqsubseteq \bigsqcap_{A' \in \Sigma \setminus \mathsf{pre}^{\Sigma}_{\mathcal{T}_0}(A)} A' \sqcap \mathsf{All}_\Sigma\Big\}$$

- If A is conjunctive in $\mathcal{T}_0$ and $A \equiv F \in \mathcal{T}_0$, then

$$\mathsf{noimply}_{\mathcal{T}_0,\Sigma}(A) = \{\xi_B \mid B \in F\} \quad \text{and} \quad \mathcal{T}_A = \emptyset$$

- If $A \equiv \exists r.B \in \mathcal{T}_0$, then $\mathsf{noimply}_{\mathcal{T}_0,\Sigma}(A) = \{\xi_A\}$ and

$$\mathcal{T}_A = \Big\{\xi_A \sqsubseteq \bigsqcap_{A' \in \Sigma \setminus \mathsf{pre}^{\Sigma}_{\mathcal{T}_0}(A)} A' \sqcap \Big(\bigsqcap_{r \neq s \in \Sigma} \exists s.\Big(\bigsqcap_{A' \in \Sigma} A' \sqcap \mathsf{All}_\Sigma\Big)\Big) \sqcap \bigsqcap_{\xi \in \mathsf{noimply}_{\mathcal{T}_0,\Sigma}(B)} \exists r.\xi\Big\}$$

Then define

$$\mathsf{Aux}_{\mathcal{T}_0,\Sigma} := \Big\{\mathsf{All}_\Sigma \sqsubseteq \bigsqcap_{r \in \Sigma} \exists r.\Big(\bigsqcap_{A \in \Sigma} A \sqcap \mathsf{All}_\Sigma\Big)\Big\} \cup \bigcup_{A \in \Sigma} \mathcal{T}_A$$

Fig. 1. Computing $\mathsf{Aux}_{\mathcal{T}_0,\Sigma}$ and $\mathsf{noimply}_{\mathcal{T}_0,\Sigma}(A)$

- $A \equiv \exists r.B$, where B is a concept name;
- $A \sqsubseteq C$, where C is a concept;
- $A \equiv F$, where F is a (possibly empty) conjunction of concept names such that every conjunct B of F is non-conjunctive.

By adapting the normalisation algorithm from [1], it is easy to show that every terminology $\mathcal{T}$ can be converted in polynomial-time into a terminology $\mathcal{T}'$ in normal form such that $\mathcal{T}' \equiv_{\mathsf{sig}(\mathcal{T})} \mathcal{T}$ and $\mathcal{T}'$ is acyclic iff $\mathcal{T}$ is acyclic. It follows that, in the proof of Theorem 1, we can w.l.o.g. assume $\mathcal{T}_0$ to be in normal form.

Lemma 2. *Let $\mathcal{T}_0$ be a terminology in normal form, Σ a signature, $\Sigma' = \{\mathsf{All}_\Sigma\} \cup \{\xi_A \mid A$ non-conjunctive in $\mathcal{T}_0\}$ a set of fresh concept names, and $\mathsf{pre}^{\Sigma}_{\mathcal{T}_0}(A) = \{B \in \Sigma \mid \mathcal{T}_0 \models B \sqsubseteq A\}$. The algorithm in Figure 1 constructs in polynomial time*

- *a terminology $\mathsf{Aux}_{\mathcal{T}_0,\Sigma}$ using only symbols from $\Sigma \cup \Sigma'$, and*
- *a set of concept names $\mathsf{noimply}_{\mathcal{T}_0,\Sigma}(A)$ for each $A \in \mathsf{sig}(\mathcal{T}_0) \cup \Sigma$*

such that the following are equivalent, for every terminology $\mathcal{T}_1$ with $\mathsf{sig}(\mathcal{T}_1) \cap \Sigma' = \emptyset$ and every $A \in \Sigma$:

1. *there exists a Σ-concept C with $\mathcal{T}_1 \models C \sqsubseteq A$ and $\mathcal{T}_0 \not\models C \sqsubseteq A$;*
2. *$\mathcal{T}_1 \cup \mathsf{Aux}_{\mathcal{T}_0,\Sigma} \models \xi \sqsubseteq A$, for some concept name $\xi \in \mathsf{noimply}_{\mathcal{T}_0,\Sigma}(A)$.*

Since subsumption in $\mathcal{EL}$ is tractable [1], Theorem 1 is an immediate consequence of Lemma 2: given $\mathcal{T}_0$, $\mathcal{T}_1$, and Σ, the following procedure runs in polynomial time: for every concept name $A \in \Sigma$, first compute $\mathsf{Aux}_{\mathcal{T}_0,\Sigma}$ and $\mathsf{noimply}_{\mathcal{T}_0,\Sigma}(A)$ and then check whether $\mathcal{T}_1 \cup \mathsf{Aux}_{\mathcal{T}_0,\Sigma} \models \xi \sqsubseteq A$, for some $\xi \in \mathsf{noimply}_{\mathcal{T}_0,\Sigma}(A)$.

In Section 5, we report about experiments with the prototype implementation CEX of a variant of this algorithm for acyclic $\mathcal{EL}$ terminologies. In contrast to the algorithm above, CEX does not proceed in two separate steps but uses a dynamic programming approach working with the two terminologies $\mathcal{T}_0$ and $\mathcal{T}_1$ at the same time.

4 Semantic modularity

We investigate the complexity of various reasoning tasks for the semantic notion of Σ-dependencies and present module extraction algorithms based on this notion. We consider the following notions of a module:

Definition 2 (Semantic modules). *Let $\mathcal{T}_0 \subseteq \mathcal{T}_1$ and $\Sigma \supseteq \mathsf{sig}(\mathcal{T}_0)$.*

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

The requirement that Σ contains the signature of $\mathcal{T}_0$ reflects the idea that modules should be self-contained in the sense that if an ontology $\mathcal{T}$ implies a dependency between symbols occuring in $\mathcal{T}_0$, then this dependency is implied by $\mathcal{T}_0$ already. Notions of modules in which this is not the case are of interest as well and are considered, for example, in [3]. The following lemma addresses the relation between strong and weak modules.

Lemma 3. *Every strong semantic module is a weak semantic module. The converse does not hold 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 imply any dependencies between symbols in Σ.

In many cases, it is much harder to decide semantic Σ-consequence than to decide Σ-entailment (which has a syntactic flavour). For general $\mathcal{EL}$-TBoxes, the first problem is undecidable while the latter is ExpTime-complete [13]. For $\mathcal{ALC}$, undecidability even applies to acyclic terminologies:

Theorem 2. *It is undecidable whether an acyclic $\mathcal{ALC}$-terminology is a semantic Σ-tautology for a signature Σ. Even for $\mathcal{ALC}$-TBoxes $\mathcal{T}$ of the form $\{A \sqsubseteq C\}$ and Σ of the form $\{A, r_1, r_2\}$, this problem is undecidable.*

This result and related ones in [7, 12] explain why the notions of modularity from Definition 2 have not yet found practical applications. Instead, applications use stronger notions such as locality [7] which, in general, leads to *larger* modules, or they use notions of modularity that are not logic-based [15, 5, 4]. In this section, we revisit (the proof of) Theorem 2 and identify relevant cases in which undecidability can be avoided. In particular, reasoning is even tractable in some of these cases. A first observation is that roles in Σ are problematic:

Theorem 3. *Let $\mathcal{L}$ be $\mathcal{ALC}$ or $\mathcal{ALCI}$. For 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 $i = 0, 1$,*

(1) it is $\text{coNExp}^{\text{NP}}$-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) it is Π_2^p-complete to decide whether $\mathcal{T}_0$ is a semantic Σ-tautology.

Thus, it is possible to decide whether a $\mathcal{T}_0 \subseteq \mathcal{T}_1$ is a weak or strong Σ-module of $\mathcal{T}_1$ in the sense of Definition 2 if Σ contains only concept names. Interestingly, deciding strong modules is considerably simpler than deciding weak modules, and even simpler than subsumption in $\mathcal{ALC}$ with acyclic TBoxes. Theorem 3 is also of interest when analysing merged ontologies: it is decidable whether the union $\mathcal{T}_0 \cup \mathcal{T}_1$ of two ontologies $\mathcal{T}_0$ and $\mathcal{T}_1$ is semantically Σ-inseparable from the two ontologies, if the set Σ of symbols shared by $\mathcal{T}_0$ and $\mathcal{T}_1$ contains only concept names.

4.1 Deciding and extracting semantic modules for acyclic terminologies

From now on, we restrict our investigation to acyclic terminologies. The following notion will play a central role:

Definition 3 (Syntactic Σ-dependency). *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.

Intuitively, having no syntactic Σ-dependency means that no concept name in Σ is defined in terms of a Σ-symbol. Syntactic Σ-dependencies give rise to a natural case in which semantic modules in $\mathcal{ALC}$ are decidable.

Theorem 4. *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 $(\Sigma, \Sigma \cap \mathsf{N_R})$-dependencies,*

(1) it is $\text{coNExp}^{\text{NP}}$-complete to decide whether $\mathcal{T}_0$ is a weak semantic Σ-module of $\mathcal{T}_1$;

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

An additional virtue of Theorem 4 is that it can be used to define approximations of a semantic module. In the following, we consider a stronger syntactic condition. Namely, we do not allow any Σ-dependency (instead of forbidding only $\Sigma \cap \mathsf{N_R}$-dependencies). Our first result states that, under the assumption that there are no syntactic Σ-dependencies, the notions of strong and weak semantic modules coincide and the corresponding decision problem is Π_2^p-complete for $\mathcal{ALC}$ and $\mathcal{ALCI}$.

Theorem 5. *Let $\mathcal{L}$ be $\mathcal{ALC}$ or $\mathcal{ALCI}$. For $\mathcal{L}$-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:*

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

1. there exists $A \in \Sigma \cap (\mathsf{Pr}(\mathcal{T}_0) \setminus \mathsf{Pr}(\mathcal{T}_1))$ with $\mathsf{depend}_{\mathcal{T}_1 \setminus \mathcal{T}_0}(A) \cap \Sigma \neq \emptyset$;
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

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

Fig. 2. Checking module in $\mathcal{ELI}$

- *$\mathcal{T}_0$ 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 $P \subseteq \Sigma \cap (\mathsf{Pr}(\mathcal{T}_0) \setminus \mathsf{Pr}(\mathcal{T}_1))$, $C_P = \bigsqcap_{A \in P} A \sqcap \bigsqcap_{A \in \Sigma \cap (\mathsf{Pr}(\mathcal{T}_0) \setminus (\mathsf{Pr}(\mathcal{T}_1) \cup P))} \neg A$ is satisfiable in a model of $\mathcal{T}_1 \setminus \mathcal{T}_0$ of cardinality 1.*

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

We now consider $\mathcal{EL}$ and $\mathcal{ELI}$. It follows from Point (2) of Theorem 2 that, in $\mathcal{ALC}$, it is undecidable whether a syntactic Σ-dependency implies that the terminology is not a semantic Σ-tautology. In $\mathcal{EL}$ and $\mathcal{ELI}$, the same problem is trivial:

Lemma 4. *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 Σ-tautology.*

Based on this observation, we show that in $\mathcal{EL}$ and $\mathcal{ELI}$, modules can be decided and extracted in polynomial-time. Say that an acyclic $\mathcal{ELI}/\mathcal{EL}$-terminology $\mathcal{T}$ *contains no trivial axioms* if no axiom of the form $A \equiv \top$ (nor $A \equiv \top \sqcap \top$, etc.) occurs in $\mathcal{T}$. Observe that the example given in the proof of Lemma 3 contains trivial axioms. In acyclic $\mathcal{ELI}$-terminologies, any A defined as $\top$ can be eliminated by replacing it with $\top$. It is, therefore, harmless to assume that such terminologies do not contain trivial axioms.

Theorem 6. *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 polynomial time whether $\mathcal{T}_0$ is a weak/strong semantic module of $\mathcal{T}_1$.

Figure 2 gives an algorithm that establishes the polynomial-time bound stated in Theorem 6, using the following notation. We say that $A \in \mathsf{N_C}$ *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 $\Sigma \supseteq \mathsf{sig}(\mathcal{T}_0)$.

Initialise: $\mathcal{T}_0 = \emptyset$.
Apply Rules 1 and 2 exhaustively, where Rule 1 has higher precedence.
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 $\mathcal{T}_0 := \mathcal{T}_0 \cup \{A \equiv C\}$.

Fig. 3. Computing module in $\mathcal{ELI}$

It is an interesting consequence of Theorem 6 that acyclic $\mathcal{ELI}$-terminologies provide the first example of a DL in which subsumption is computationally harder than checking semantic modules: the former is PSPACE-complete [9], and the latter can be done in polynomial-time. We now apply the results on checking semantic modules above to design a module extraction algorithm. For $\mathcal{EL}$ and $\mathcal{ELI}$, the algorithm is presented in Figure 3. It takes as input an acyclic $\mathcal{ELI}$-terminology $\mathcal{T}_1$ containing no trivial axioms and a signature Σ and it outputs $\mathcal{T}_0$.

Theorem 7. *Let $\mathcal{L}$ be $\mathcal{EL}$ or $\mathcal{ELI}$. For an acyclic $\mathcal{L}$-terminology $\mathcal{T}_1$ containing no trivial axioms and a signature Σ, the algorithm in Figure 3 outputs the smallest $\mathcal{T}_0 \subseteq \mathcal{T}_1$ such that $\mathcal{T}_0$ is a strong/weak semantic $\Sigma \cup \mathsf{sig}(\mathcal{T}_0)$-module of $\mathcal{T}_1$.*

For module extraction algorithms for $\mathcal{ALC}$ and $\mathcal{ALCI}$, we refer to the full paper [10].

5 CEX and MEX—Experimental results

For any two acyclic $\mathcal{EL}$-terminologies $\mathcal{T}_0$ and $\mathcal{T}_1$ and signature Σ, the system CEX outputs two lists $\mathsf{DiffR}_\Sigma(\mathcal{T}_0, \mathcal{T}_1)$ and $\mathsf{DiffL}_\Sigma(\mathcal{T}_0, \mathcal{T}_1)$ that describe the logical difference between $\mathcal{T}_0$ and $\mathcal{T}_1$. Formally,

- the list $\mathsf{DiffR}_\Sigma(\mathcal{T}_0, \mathcal{T}_1)$ consists of all $A \in \Sigma$ such that there is a Σ-concept C with $\mathcal{T}_0 \not\models C \sqsubseteq A$ and $\mathcal{T}_1 \models C \sqsubseteq A$.
- the list $\mathsf{DiffL}_\Sigma(\mathcal{T}_0, \mathcal{T}_1)$ consists of all $A \in \Sigma$ such that there is a Σ-concept C with $\mathcal{T}_0 \not\models A \sqsubseteq C$ and $\mathcal{T}_1 \models A \sqsubseteq C$.

Both lists are empty if, and only if, $\mathcal{T}_0$ and $\mathcal{T}_1$ are Σ-inseparable w.r.t. $\mathcal{EL}$.

Given an acyclic $\mathcal{EL}$-terminology $\mathcal{T}_1$ and signature Σ, the system MEX outputs the smallest semantic module $\mathcal{T}_0$ of $\mathcal{T}_1$ as described in Theorem 7. MEX and CEX are OCaml programs. The experiments below use two versions of SNOMED

CT: one dated 09 February 2005 (SM-05) and the other 30 December 2006 (SM-06) and having 379 691 and 389 472 axioms, respectively. As CEX currently accepts acyclic $\mathcal{EL}$-terminologies only, the role inclusions of SNOMED CT are not taken into account for experiments with CEX (in contrast to MEX, see [10] for details). The tests have been carried out on a standard PC: Intel® Core™ 2 CPU at 2.13 GHz and 3 GB of RAM.

Logical difference between SM-05 *and* SM-06. The left hand side of Table 1 shows the average time and memory consumption of CEX computing the lists $\mathsf{DiffR}_\Sigma(\text{SM-05}, \text{SM-06})$ and $\mathsf{DiffL}_\Sigma(\text{SM-05}, \text{SM-06})$ for 20 randomly generated signatures Σ of size 100, 1 000, etc. For both sets, their average size is provided.

The right hand side of Table 1 shows the average time and memory consumption of computing *the same lists*, but here we first use MEX to extract semantic Σ-modules $\mathcal{T}_0$ and $\mathcal{T}_1$ from SM-05 and SM-06, respectively, and then CEX computes $\mathsf{DiffR}_\Sigma(\mathcal{T}_0, \mathcal{T}_1)$ and $\mathsf{DiffL}_\Sigma(\mathcal{T}_0, \mathcal{T}_1)$. Though CEX is already very efficient, the results show that the latter procedure is even faster and gives results almost instantaniously. Even in the case of $\Sigma = 100000$, MEX contributes less than 5 seconds to the overall runtime.

Size of Σ	CEX: Diff(SM-05,SM-06)				CEX: Diff(Mod(SM-05),Mod(SM-06))	
	Time (Sec.)	Memory (MByte)	$\lvert\mathsf{DiffL}_\Sigma\rvert$	$\lvert\mathsf{DiffR}_\Sigma\rvert$	Time (Sec.)	Memory (MByte)
100	513.1	1 393.7	0.0	0.0	3.66	116.5
1 000	512.4	1 394.6	2.5	2.5	4.46	122.5
10 000	517.7	1 424.3	183.2	122.0	22.29	126.5
100 000	559.8	1 473.2	11 322.1	4 108.5	189.98	615.8

Table 1. Logical difference between semantic modules of two SNOMED CT versions

The size of modules. We compare the size of modules generated by MEX with the minimal modules generated by a number of other extraction algorithms. When applied to an acyclic $\mathcal{EL}$-terminology $\mathcal{T}_1$ and signature Σ, the module extraction algorithms presented in [7, 5, 15] output a module $\mathcal{T}_0$ that is *definition-closed*, i.e., satisfies the following:

if $A \in \mathsf{sig}(\mathcal{T}_0) \cup \Sigma$ and $\alpha \in \mathcal{T}_1$ has A on the left hand side, then $\alpha \in \mathcal{T}_0$.

An exception are the modules generated using the $\top$-based locality approach of [7] (whereas the $\bot$-based locality approach yields definition-closed modules). Any definition-closed module contains the module generated by MEX. The following experiment compares the minimal size of definition-closed modules with the size of modules generated using MEX, when applied to SNOMED CT. To compute the minimal definition-closed modules, we use the module extraction

feature of the CEL reasoner [2] (Version 1.0b). In Figure 4, an input signature consisted of a number of concept names that were randomly selected from SM-05. The size of the input signatures varied from 100 to 1 000 concept names and for every signature size, we use 1000 random signatures. The maximal, minimal, and average module sizes depending on the size of the input signature are shown.

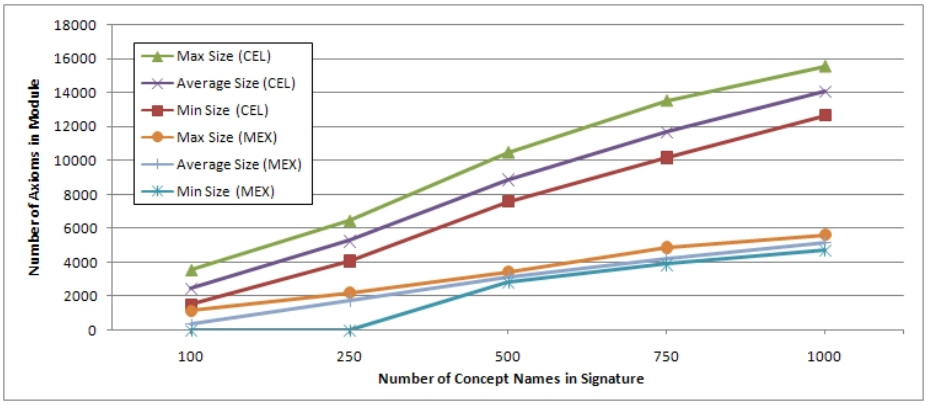

Fig. 4. Sizes of definition-closed and semantic modules

Comparing sensitivity of CEX and class hierarchy. To compute the logical diff, CEX compares the set of Σ-concept inclusions implied by terminologies. Alternatively, one could compare only the set of implied inclusions between Σ-concept names (the class-hierarchy). Figure 5 shows the size of $\mathsf{DiffL}_\Sigma \cup \mathsf{DiffR}_\Sigma$ as computed by CEX when applied to the empty terminology and SM-05 for 500 randomly generated signatures Σ of size 10 to 270. Each Σ contains, in addition, 20 role names. The second curve of Figure 5 shows the number of concept names from Σ for which new inclusions occur in the class-hierarchy restricted to Σ. It can be seen that for these signatures CEX detects about five times as many differences as the class hierarchy. The difference between the two approaches is less significant if less roles names are in the signature Σ. But even for signatures without role names CEX detects possibly important differences that do not occur in the class hierarchy.

For further experiments using CEX and MEX, we refer the reader to the full papers [11, 10].

References

1. F. Baader. Terminological cycles in a description logic with existential restrictions. In *Proc. of IJCAI-03*, pages 325–330. Morgan Kaufmann, 2003. Long version available as LTCS Report 02-02.
2. F. Baader, C. Lutz, and B. Suntisrivaraporn. CEL—a polynomial-time reasoner for life science ontologies. In *Proc. of IJCAR-06*, volume 4130 of *LNAI*, pages 287–291. Springer, 2006.

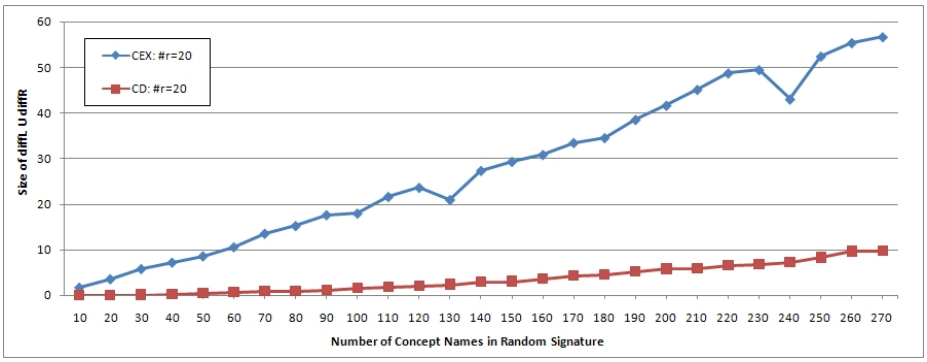

Fig. 5. Comparing sensitivity of CEX and class hierarchy

3. A. Borgida. On importing knowdledge from DL ontologies: some intuitions and problems. In *Proc. of DL Workshop*, volume 250 of http://ceur-ws.org/, 2007.
4. P. Doran, V. Tamma, and L.Iannone. Ontology module extraction for ontology reuse: an ontology engineering perspective. In *Proc. of CIKM-07*, pages 61–70. ACM, 2007.
5. J. H. Gennari *et al.* The evolution of protégé: an environment for knowledge-based systems development. *Int. J. Hum.-Comput. Stud.*, 58(1):89–123, 2003.
6. S. Ghilardi, C. Lutz, and F. Wolter. Did I damage my ontology? a case for conservative extensions in description logics. In *Proc. of KR-06*, pages 187–197. AAAI Press, 2006.
7. B. C. Grau, I. Horrocks, Y. Kazakov, and U. Sattler. Just the right amount: extracting modules from ontologies. In *Proc. of WWW-07*, pages 717–726, 2007.
8. B. C. Grau, I. Horrocks, Y. Kazakov, and U. Sattler. A logical framework for modularity of ontologies. In *Proc. of IJCAI-07*, pages 298–303. AAAI Press, 2007.
9. C. Haase and C. Lutz. Complexity of subsumption in the $\mathcal{EL}$-family of description logics: Acyclic and cyclic Tboxes. 2008. submitted.
10. B. Konev, C. Lutz, D. Walther, and F. Wolter. Semantic modularity and module extraction in description logic. `http://www.csc.liv.ac.uk/frank/publ/publ.html`.
11. B. Konev, D. Walther, and F. Wolter. The logical difference problem for description logic terminologies. `http://www.csc.liv.ac.uk/frank/publ/publ.html`.
12. C. Lutz, D. Walther, and F. Wolter. Conservative extensions in expressive description logics. In *Proc. of IJCAI-07*, pages 453–458. AAAI Press, 2007.
13. C. Lutz and F. Wolter. Conservative extensions in the lightweight description logic $\mathcal{EL}$. In *Proc. of CADE-07*, volume 4603 of LNCS, pages 84–99. Springer, 2007.
14. N. F. Noy and M. Musen. Promptdiff: A fixed-point algorithm for comparing ontology versions. In *Proc. of AAAI*, pages 744–750. AAAI Press, 2002.
15. J. Seidenberg and A. L. Rector. Web ontology segmentation: analysis, classification and use. In *WWW-06*, pages 13–22, ACM, 2006.
16. K. Spackman. Managing clinical terminology hierarchies using algorithmic calculation of subsumption: Experience with SNOMED-RT. In *JAMIA*, 2000.
17. B. Suntisrivaraporn. Module Extraction and Incremental Classification: A Pragmatic Approach for $\mathcal{EL}^+$ Ontologies. In *Proc. of ESWC-08*, volume 5021 of LNCS, pages 230–244. Springer, 2008.