On Computing Minimal $\mathscr{EL}$-Subsumption Modules

Jieying CHEN [a], Michel LUDWIG [b], and Dirk WALTHER [b]

[a] *Laboratoire de Recherche en Informatique, Université Paris-Sud, France*
jieying.chen@lri.fr

[b] *Theoretical Computer Science, TU Dresden, Germany*
$\{$*michel,dirk*$\}$*@tcs.inf.tu-dresden.de*

Abstract. In the paper we study algorithms for computing minimal modules that are minimal w.r.t. set inclusion and that preserve the entailment of all $\mathscr{EL}$-subsumptions over a signature of interest. We follow the black-box approach for finding one or all justifications by replacing the entailment tests with logical difference checks, obtaining modules that preserve not only a given consequence but all entailments over a signature. Such minimal modules can serve to improve our understanding of the internal structure of large and complex ontologies. Additionally, several optimisations to speed up the computation of minimal modules are investigated. We present an experimental evaluation of an implementation of our algorithms by applying them on the medical ontologies Snomed CT and NCI.

1. Introduction

A module is a subset of an ontology that can act as a substitute for the ontology in certain contexts. A basic requirement on modules is to be indistinguishable from the original ontology w.r.t. an inseparability relation. Such *basic* modules are also called ‘plain’ modules. Further module properties such as *self-containment* and *depletion* have been proposed in the literature [9, 11] (also called *weak* and *strong* in [8]). These properties together with inseparability relations give rise to a family of module notions. Several inseparability notions have been considered, e.g., model-theoretic inseparability w.r.t. a signature [9], or inseparability w.r.t. answers to queries [13] (i.e., two ontologies are inseparable iff they entail the same queries). Popular query types are subsumption, instance and conjunctive queries. In particular, for $\mathscr{EL}$-TBoxes model-theoretic inseparability w.r.t. a signature Σ coincides with entailment of second-order sentences over Σ (cf. Theorem 4 in [9]). We call modules based on model-theoretic inseparability *semantic modules*. In this paper, however, we consider a weaker inseparability relation that is based on subsumption queries between $\mathscr{EL}$-concepts over a given signature. We call the resulting modules $\mathscr{EL}$*-subsumption modules*.

An important requirement on modules is that they should be as small as possible, which is particularly useful, e.g., in the ontology re-use scenario [4]. As smallest modules are not necessarily unique, we are interested in computing all basic $\mathscr{EL}$-subsumption modules that are minimal w.r.t. set inclusion. Computing minimal basic semantic modules of $\mathcal{EL}$-terminologies that are additionally self-contained and depleting has been investigated in [8, 9]. Algorithms for computing minimal modules of DL-Lite ontologies have been studied in [11]. However, to the best of our knowledge, no practical approach for computing one or all basic $\mathcal{EL}$-subsumption modules of $\mathcal{EL}$-terminologies has been considered.

Minimal modules can serve as explanations of the entire set of entailments over a signature, similar to the justifications for one consequence (i.e., minimal sets of axioms sufficient to entail the consequence) [5]. In this sense, minimal modules can improve our understanding of the internal structure of large and complex ontologies. Moreover, being able to compute all minimal modules allows us to select the smallest minimal module.

In general, extracting minimal modules is intractable, which is the reason why efficiently extractable approximations of the (union of all) minimal modules have been introduced. Among such approximations are the family of syntactic locality-based modules [4]. Such modules may contain more axioms than necessary to ensure the preservation of entailments over a signature. For instance, the size of the syntactic $\bot\top^*$-locality modules [16] of Snomed CT (version Jan 2016) for 100 signatures consisting of 50 concept names selected at random together with all roles names ranges from 1 075 to 2 456 axioms. This is in contrast to the size of the minimal basic $\mathcal{EL}$-subsumption modules for these signatures that ranges from around 50 to 118 axioms. Hence, such minimal modules of Snomed CT may be more than 20 times smaller than the corresponding syntactic $\bot\top^*$-locality modules.

The system MEX has been introduced to compute minimal depleting semantic modules (which are unique for a given signature) from acyclic $\mathcal{EL}$-terminologies (possibly extended with inverse roles) such as Snomed CT. The MEX-modules contain all minimal basic $\mathcal{EL}$-subsumption modules, i.e., the module notion that we are interested in. The size of the MEX-modules of Snomed CT for the same signatures as above ranges from 401 to 720 axioms. However, the corresponding minimal basic $\mathcal{EL}$-subsumption modules are still at least 6 times smaller. Moreover, MEX cannot handle cyclic $\mathcal{EL}$-terminologies such as some recent versions of NCI. For instance, the size of the syntactic $\bot\top^*$-locality module of NCI (version 14.01d) for 100 random signatures selected from NCI (as before for Snomed CT just with 100 concept names) ranges from 679 to 3 895 axioms, whereas the size of the corresponding minimal basic $\mathcal{EL}$-subsumption modules ranges from around 0 to 64 axioms. Clearly, the ratio of the size of the syntactic $\bot\top^*$-locality based modules compared to the size of the minimal basic $\mathcal{EL}$-subsumption modules is even larger than 20 in this case. Another approach for extracting minimal depleting modules from DL-Lite ontologies is based on using QBF-solvers [11].

In this paper, in order to compute minimal basic $\mathcal{EL}$-subsumption modules we extend the black-box approach for finding one or all justifications in [5], which is based on Reiter's hitting set algorithm [15]. Instead of ensuring that a given entailment is preserved, we introduce an oracle to determine the inseparability between the original ontology and the resulting module. As an oracle we use a variant of the system CEX, which is the only currently available tool for deciding whether two $\mathcal{EL}$-terminologies are logically different w.r.t. a signature [7, 10]. Additionally, several optimisations to speed up the computation of minimal modules are investigated. We present an experimental evaluation of our algorithms by applying them on the prominent and large medical ontologies Snomed CT and NCI. We note that our algorithms are applicable to ontologies formulated in any ontology language provided that a tool is available that can effectively decide the inseparability relation. As CEX works with variants of $\mathcal{EL}$-terminologies, we restrict the presentation of our algorithms to $\mathcal{EL}$-terminologies.

We proceed as follows. We start by reviewing $\mathcal{EL}$-terminologies together with the notion of logical difference. In Section 3 we define the notion of basic $\mathcal{EL}$-subsumption module and we introduce algorithms for extracting one or all minimal such modules. In Section 4 we present the results of an evaluation of our algorithms using Snomed CT and NCI. We close the paper with a conclusion.

2. Preliminaries

Let $\mathsf{N_C}$ and $\mathsf{N_R}$ be mutually disjoint (countably infinite) sets of concept names and role names. In the following we use upper case letters A, B, X, Y, Z to denote concept names, and lower case letters r, s stand for role names. The set of *$\mathcal{EL}$-concepts* C and the set of *$\mathcal{EL}$-inclusions* α are built according to the following grammar rules: $C ::= \top \mid A \mid C \sqcap C \mid \exists r.C$ and $\alpha ::= C \sqsubseteq C \mid C \equiv C$, where $A \in \mathsf{N_C}$ and $r, s \in \mathsf{N_R}$. An *$\mathcal{EL}$-TBox* $\mathcal{T}$ is a finite set of $\mathcal{EL}$-inclusions. We also refer to $\mathcal{EL}$-inclusions as *axioms* when they are contained in an $\mathcal{EL}$-TBox.

The semantics is defined using interpretations $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where the 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 every role name r to a binary relation $r^{\mathcal{I}}$ over $\Delta^{\mathcal{I}}$. The *extension* $C^{\mathcal{I}}$ of a possibly complex concept C is defined inductively as: $(\top)^{\mathcal{I}} := \Delta^{\mathcal{I}}$, $(C \sqcap D)^{\mathcal{I}} := C^{\mathcal{I}} \cap D^{\mathcal{I}}$, and $(\exists r.C)^{\mathcal{I}} := \{x \in \Delta^{\mathcal{I}} \mid \exists y \in C^{\mathcal{I}} : (x,y) \in r^{\mathcal{I}}\}$.

An interpretation $\mathcal{I}$ *satisfies* an $\mathcal{EL}$-concept C, an $\mathcal{EL}$-inclusion $C \sqsubseteq D$, or $C \equiv D$ if $C^{\mathcal{I}} \neq \emptyset$, $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, or $C^{\mathcal{I}} = D^{\mathcal{I}}$, respectively. We write $\mathcal{I} \models \alpha$ if $\mathcal{I}$ satisfies the axiom α. Note that every $\mathcal{EL}$-concept and $\mathcal{EL}$-inclusion is satisfiable, but a particular interpretation does not necessarily satisfy a concept or inclusion. An interpretation $\mathcal{I}$ is a *model* of $\mathcal{T}$ if $\mathcal{I}$ satisfies all axioms in $\mathcal{T}$. An $\mathcal{EL}$-inclusion α *follows* from an $\mathcal{EL}$-TBox $\mathcal{T}$, written $\mathcal{T} \models \alpha$, if for all models $\mathcal{I}$ of $\mathcal{T}$, we have that $\mathcal{I} \models \alpha$.

A signature Σ is a finite set of symbols from $\mathsf{N_C}$ and $\mathsf{N_R}$. The signature $\mathsf{sig}(\varphi)$ is the set of concept and role names occurring in φ, where φ ranges over any syntactic object. We set $\mathsf{sig}^{\mathsf{N_C}}(\varphi) := \mathsf{sig}(\varphi) \cap \mathsf{N_C}$. The symbol Σ is used as a subscript to a set of concepts or axioms to denote that the elements only use symbols from Σ, e.g., $\mathcal{EL}_\Sigma$, etc.

An $\mathcal{EL}$-terminology $\mathcal{T}$ is an $\mathcal{EL}$-TBox consisting of axioms of the forms $X \sqsubseteq C$ or $X \equiv C$, where X is a concept name in $\mathsf{N_C}$ and C is an $\mathcal{EL}$-concept, and no concept name X occurs more than once on the left-hand side of an axiom. A terminology is said to be acyclic if it can be unfolded (i.e., the process of substituting concept names by the right-hand sides of their defining axioms terminates). Formally, we define the relation $\prec_{\mathcal{T}}$: $\mathsf{sig}^{\mathsf{N_C}}(\mathcal{T}) \times \mathsf{sig}^{\mathsf{N_C}}(\mathcal{T})$ by setting $(X,Y) \in \prec_{\mathcal{T}}$ iff there exists an axiom of the form $X \equiv C$ or $X \sqsubseteq C$ in $\mathcal{T}$ such that $Y \in \mathsf{sig}(C)$. Then, a terminology $\mathcal{T}$ is acyclic iff the transitive closure $(\prec_{\mathcal{T}})^+$ of $\prec_{\mathcal{T}}$ is irreflexive. For instance, the prominent medical ontology Snomed CT (version Jan 2016) is an acyclic $\mathcal{EL}$-terminology, whereas the ontology NCI (version 14.01d) is a cyclic $\mathcal{EL}$-terminology.

We now recall basic notions related to the logical difference between two $\mathcal{EL}$-terminologies for $\mathcal{EL}$-inclusions over a given signature as query language [6, 10].

Definition 1 (Logical Difference) *Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be two $\mathcal{EL}$-terminologies, and let Σ be a signature. The $\mathcal{EL}$-*concept inclusion difference *between $\mathcal{T}_1$ and $\mathcal{T}_2$ w.r.t. Σ is the* *set* $\mathsf{cDiff}_\Sigma(\mathcal{T}_1, \mathcal{T}_2)$ *of all* $\mathcal{EL}$*-inclusions* α *of the form* $C \sqsubseteq D$ *for* $\mathcal{EL}$*-concepts* C *and* D *such that* $\mathsf{sig}(\alpha) \subseteq \Sigma$, $\mathcal{T}_1 \models \alpha$, *and* $\mathcal{T}_2 \not\models \alpha$.

Two $\mathcal{EL}$-terminologies $\mathcal{T}_1$ and $\mathcal{T}_2$ are also called *inseparable w.r.t.* $\mathcal{EL}$*-concept inclusions over* Σ iff $\mathsf{cDiff}_\Sigma(\mathcal{T}_1, \mathcal{T}_2) = \emptyset$ and $\mathsf{cDiff}_\Sigma(\mathcal{T}_2, \mathcal{T}_1) = \emptyset$ [13]. If there exists an $\mathcal{EL}$-inclusion α such that $\mathsf{sig}(\alpha) \subseteq \Sigma$, $\mathcal{T}_1 \models \alpha$, and $\mathcal{T}_2 \not\models \alpha$, then the set $\mathsf{cDiff}_\Sigma(\mathcal{T}_1, \mathcal{T}_2)$ consists of infinitely many concept inclusions.

For acyclic $\mathcal{EL}$-terminologies $\mathcal{T}_1$ and $\mathcal{T}_2$, the version 2.5 of the system CEX [7,10] can decide whether the set $\mathsf{cDiff}_\Sigma(\mathcal{T}_1, \mathcal{T}_2)$ is empty. In this paper we use a variant of CEX that works with cyclic $\mathcal{EL}$-terminologies, implementing a hypergraph-based approach to the logical difference problem as introduced in [3] and further extended in [12].

3. Minimal Modules

We now give a formal definition of the module notion that we consider in this paper.

Definition 2 (Basic $\mathcal{EL}$-Subsumption Module) *Let* $\mathcal{T}$ *be an* $\mathcal{EL}$*-terminology, and let* Σ *be a signature. A subset* $\mathcal{M} \subseteq \mathcal{T}$ *is called a* basic $\mathcal{EL}$-subsumption module of $\mathcal{T}$ w.r.t. Σ *iff for all* $\mathcal{EL}$*-inclusions* α *of the form* $C \sqsubseteq D$ *for* $\mathcal{EL}$*-concepts* C *and* D *with* $\mathsf{sig}(\alpha) \subseteq \Sigma$ *it holds that* $\mathcal{T} \models \alpha$ *iff* $\mathcal{M} \models \alpha$.

Every subset $\mathcal{M}$ of a terminology $\mathcal{T}$ that preserves the entailment of all $\mathcal{EL}$-subsumptions over a given signature Σ is a basic $\mathcal{EL}$-subsumption module of $\mathcal{T}$ w.r.t. Σ. In particular, $\mathcal{T}$ itself is a basic $\mathcal{EL}$-subsumption module of $\mathcal{T}$ w.r.t. any signature. It can readily be seen that $\mathcal{M}$ is a basic $\mathcal{EL}$-subsumption module of $\mathcal{T}$ w.r.t. Σ iff $\mathsf{cDiff}_\Sigma(\mathcal{T}, \mathcal{M}) = \emptyset$ (cf. Definition 1). We have that $\mathcal{M}$ and $\mathcal{T}$ are inseparable w.r.t. $\mathcal{EL}$-inclusions over Σ. More precisely, as $\mathcal{M} \subseteq \mathcal{T}$, it holds that $\mathcal{T}$ is a conservative extension of $\mathcal{M}$ w.r.t. $\mathcal{EL}$-inclusions over Σ [13]. There may be exponentially many (in the size of $\mathcal{T}$) subsets of $\mathcal{T}$ that satisfy that criterion (see Example 6). For the use-case of ontology re-use, however, we are most interested in modules that are as small as possible [4]. Note that smallest modules (regarding the number of axioms) are also minimal w.r.t. $\subseteq$, whereas the converse does not hold in general, i.e., there may be minimal modules w.r.t. $\subseteq$ that contain more axioms than other minimal modules w.r.t. $\subseteq$.

Example 3 *Let* $\mathcal{T} = \{A \sqsubseteq X \sqcap Y, X \sqsubseteq B, Y \sqsubseteq Z, Z \sqsubseteq B\}$ *be an* $\mathcal{EL}$*-terminology, and* $\Sigma = \{A, B\}$ *be a signature. It holds that both sets,* $\mathcal{M}_1 = \{A \sqsubseteq X \sqcap Y, X \sqsubseteq B\}$ *and* $\mathcal{M}_2 = \{A \sqsubseteq X \sqcap Y, Y \sqsubseteq Z, Z \sqsubseteq B\}$, *are minimal basic* $\mathcal{EL}$*-subsumption modules of* $\mathcal{T}$ *w.r.t.* Σ, *whereas* $\mathcal{M}_1$ *is the smallest minimal basic* $\mathcal{EL}$*-subsumption module of* $\mathcal{T}$ *w.r.t.* Σ *as* $|\mathcal{M}_1| < |\mathcal{M}_2|$.

The notion of a justification for a concept inclusion α has been introduced as a minimal subset of a TBox that entails a given concept inclusion [2]. We can understand a minimal module as a more general notion of justification: a minimal basic $\mathcal{EL}$-subsumption module of $\mathcal{T}$ w.r.t. Σ is a justification for all the concept inclusions over Σ entailed by $\mathcal{T}$.

Semantic modules of $\mathcal{EL}$-terminologies that are self-contained or depleting (in fact, such modules have both properties [9]) can be larger than basic $\mathcal{EL}$-subsumption modules as introduced in Definition 2.

Example 4 *Let $\mathcal{T} = \{A \sqsubseteq \exists r.B\}$ be an $\mathcal{EL}$-terminology, and $\Sigma = \{A, B\}$ be a signature. It is easy to verify that $\mathcal{T}$ itself is a basic, self-contained, and depleting semantic module of $\mathcal{T}$ w.r.t. Σ [8,9], whereas the empty set is the minimal basic $\mathcal{EL}$-subsumption module of $\mathcal{T}$ w.r.t. Σ.*

The following example extends Example 3 to show that the modules computed by the system MEX [8] as well as the modules based on syntactic locality can be larger than minimal basic $\mathcal{EL}$-subsumption modules as introduced in Definition 2. Note that MEX-modules are semantic modules that are self-contained as well as depleting [9] (equivalently, weak and strong [8]).

Example 5 *Let $\mathcal{T} = \{A \sqsubseteq X \sqcap Y \sqcap U, X \sqsubseteq B, Y \sqsubseteq Z, Z \sqsubseteq B, U \equiv V \sqcap W\}$ be an $\mathcal{EL}$-terminology, and $\Sigma = \{A, B\}$ be a signature. It holds that both sets, $\mathcal{M}_1 = \{A \sqsubseteq X \sqcap Y, X \sqsubseteq B\}$ and $\mathcal{M}_2 = \{A \sqsubseteq X \sqcap Y, Y \sqsubseteq Z, Z \sqsubseteq B\}$, are minimal basic $\mathcal{EL}$-subsumption modules of $\mathcal{T}$ w.r.t. Σ. Moreover, MEX outputs $\mathcal{M}_3 = \mathcal{M}_1 \cup \mathcal{M}_2 = \{A \sqsubseteq X \sqcap Y \sqcap U, X \sqsubseteq B, Y \sqsubseteq Z, Z \sqsubseteq B\}$ as module of $\mathcal{T}$ w.r.t. Σ. Finally, $\mathcal{T}$ itself is the $\bot\top^*$-local module of $\mathcal{T}$ w.r.t. Σ.*

In general, there can be several minimal basic $\mathcal{EL}$-subsumption modules of an acyclic $\mathcal{EL}$-terminology for a signature, and even the smallest of such modules are not necessarily unique. The next example shows a sequence of acyclic $\mathcal{EL}$-terminologies whose number of minimal basic $\mathcal{EL}$-subsumption modules for a given signature is exponentially increasing.

Example 6 *Let $\mathcal{T}_n = \{A \sqsubseteq X_0\} \cup \{X_{i-1} \sqsubseteq Y_i \sqcap Z_i \mid 1 \leq i \leq n\} \cup \{Y_i \sqsubseteq X_i, Z_i \sqsubseteq X_i \mid 1 \leq i \leq n\} \cup \{X_n \sqsubseteq B\}$ with $n \geq 0$ be $\mathcal{EL}$-terminologies, and let $\Sigma = \{A, B\}$ be a signature.*

It holds that the set $\{A \sqsubseteq X_0, X_0 \sqsubseteq B\}$ is the minimal basic $\mathcal{EL}$-subsumption module of $\mathcal{T}_0$ w.r.t. Σ, the sets $\{A \sqsubseteq X_0, X_0 \sqsubseteq Y_1 \sqcap Z_1, Y_1 \sqsubseteq X_1, X_1 \sqsubseteq B\}$ and $\{A \sqsubseteq X_0, X_0 \sqsubseteq Y_1 \sqcap Z_1, Z_1 \sqsubseteq X_1, X_1 \sqsubseteq B\}$ are the two minimal basic $\mathcal{EL}$-subsumption modules of $\mathcal{T}_1$, and the sets $\{A \sqsubseteq X_0, X_0 \sqsubseteq Y_1 \sqcap Z_1, Y_1 \sqsubseteq X_1, X_1 \sqsubseteq Y_2 \sqcap Z_2, Y_2 \sqsubseteq X_2, X_2 \sqsubseteq B\}$, $\{A \sqsubseteq X_0, X_0 \sqsubseteq Y_1 \sqcap Z_1, Y_1 \sqsubseteq X_1, X_1 \sqsubseteq Y_2 \sqcap Z_2, Z_2 \sqsubseteq X_2, X_2 \sqsubseteq B\}$, $\{A \sqsubseteq X_0, X_0 \sqsubseteq Y_1 \sqcap Z_1, Z_1 \sqsubseteq X_1, X_1 \sqsubseteq Y_2 \sqcap Z_2, Y_2 \sqsubseteq X_2, X_2 \sqsubseteq B\}$, and $\{A \sqsubseteq X_0, X_0 \sqsubseteq Y_1 \sqcap Z_1, Z_1 \sqsubseteq X_1, X_1 \sqsubseteq Y_2 \sqcap Z_2, Z_2 \sqsubseteq X_2, X_2 \sqsubseteq B\}$ are the four minimal basic $\mathcal{EL}$-subsumption modules of $\mathcal{T}_2$, etc. In general, it can readily be verified that $\mathcal{T}_n$ has 2^n many distinct minimal basic $\mathcal{EL}$-subsumption modules w.r.t. Σ.

In the remainder of this section, we present algorithms for computing minimal basic $\mathcal{EL}$-subsumption modules. In Section 4 we analyse the number of minimal basic $\mathcal{EL}$-subsumption modules in large medical ontologies for certain signatures. We will simply write *module* instead of 'basic $\mathcal{EL}$-subsumption module'.

3.1. Computing a Single Minimal Module

A first straightforward procedure SINGLE-MINIMAL-MODULE for computing a minimal module of an $\mathcal{EL}$-terminology $\mathcal{T}$ w.r.t. a signature Σ is given in Algorithm 1.[1] The procedure operates as follows. First, the variable $\mathcal{M}$ is initialised with $\mathcal{T}$. Subsequently,

[1]A similar algorithm for DL-Lite ontologies has already been described in Theorem 67 of [11].

the procedure iterates over every axiom $\alpha \in \mathscr{T}$ and checks whether $\mathsf{cDiff}_\Sigma(\mathscr{T}, \mathscr{M} \setminus \{\alpha\}) = \emptyset$, in which case the axiom α is removed from $\mathscr{M}$. During the execution of the while-loop the set $\mathscr{M}$ is hence shrunk by removing axioms that do not lead to a logical difference until a minimal module of $\mathscr{T}$ for Σ remains.

Algorithm 1 Computing a Single Minimal Module w.r.t. a Signature

```
INPUT: $\mathscr{EL}$-terminology $\mathscr{T}$, signature $\Sigma$
1: function SINGLE-MINIMAL-MODULE($\mathscr{T}$,$\Sigma$)
2:     $\mathscr{M} := \mathscr{T}$
3:     for every axiom $\alpha \in \mathscr{T}$ do
4:         if $\mathsf{cDiff}_\Sigma(\mathscr{T}, \mathscr{M} \setminus \{\alpha\}) = \emptyset$ then
5:             $\mathscr{M} := \mathscr{M} \setminus \{\alpha\}$
6:         end if
7:     end for
8:     return $\mathscr{M}$
9: end function
```

Note that the minimal module that is extracted by Algorithm 1 depends on the order in which axioms were chosen during the iteration (Line 3), i.e. by iterating over the axioms in a different order one can potentially obtain a different minimal module. Moreover, one can show that all minimal modules can be computed by using all possible orderings on the axioms $\alpha \in \mathscr{T}$ in the for-loop in Line 3.

It is easy to see that Algorithm 1 always terminates and that it runs in polynomial time in the size of $\mathscr{T}$ and Σ since deciding the existence of a logical difference between $\mathscr{EL}$-terminologies can be performed in polynomial time in the size of $\mathscr{T}$ and Σ.

Regarding correctness, if we assume towards a contradiction that a set $\mathscr{M}_{\text{min}} \subseteq \mathscr{T}$ computed by Algorithm 1 applied on $\mathscr{T}$ and Σ is not a minimal module of $\mathscr{T}$ w.r.t. Σ, then there would exist an axiom $\alpha \in \mathscr{M}$ such that $\mathsf{cDiff}_\Sigma(\mathscr{T}, \mathscr{M}_{\text{min}} \setminus \{\alpha\}) = \emptyset$. However, when α was analysed in the for-loop in Line 3, $\mathsf{cDiff}_\Sigma(\mathscr{T}, \mathscr{M}' \setminus \{\alpha\})$ must have been empty as well by monotonicity of $\models$, where $\mathscr{M}'$ with $\mathscr{M}_{\text{min}} \subseteq \mathscr{M}'$ represents the value of the variable $\mathscr{M}$ in Algorithm 1 at the time α was inspected. Consequently, it would hold that $\alpha \notin \mathscr{M}_{\text{min}}$ and we have derived a contradiction. We hence obtain the following result.

Theorem 7 *Let $\mathscr{T}$ be an $\mathscr{EL}$-terminology and let Σ be a signature. Then Algorithm 1 applied on $\mathscr{T}$ and Σ computes a minimal module of $\mathscr{T}$ for Σ.*

As checking the existence of a logical difference can be costly in practice, we now introduce a refinement of the previous algorithm that potentially allows it to reduce the number of logical difference checks that are required for computing a minimal module. The refined procedure SINGLE-MINIMAL-MODULE-BUBBLE is shown in Algorithm 2.

Intuitively, instead of checking whether the removal of a single axiom leads to a logical difference, the refined procedure removes a set $\mathscr{B}$ of axioms from $\mathscr{T}$ at once. Such a set $\mathscr{B}$ is also called a *bubble*. As an additional optimisation we introduce the notion of logical difference core, which will become relevant in the context of computing all minimal modules when the algorithm for computing one minimal module has to be executed frequently.

Algorithm 2 Computing a Single Minimal Module w.r.t. a Signature using Axiom Bubbles

```
INPUT: $\mathcal{EL}$-terminology $\mathcal{T}$, signature $\Sigma$, $n \geq 1$, logical difference core $\mathcal{C} \subseteq \mathcal{T}$ w.r.t. $\Sigma$
 1: function SINGLE-MINIMAL-MODULE-BUBBLE($\mathcal{T}, \Sigma, n, \mathcal{C}$)
 2:     $\mathcal{M} := \mathcal{T}$
 3:     $\mathbb{Q} :=$ SPLIT($\mathcal{T} \setminus \mathcal{C}, n$)
 4:     while $\mathbb{Q} \neq []$ do
 5:         $\mathcal{B} :=$ HEAD($\mathbb{Q}$)
 6:         $\mathbb{Q} :=$ TAIL($\mathbb{Q}$)
 7:         if $\mathsf{cDiff}_\Sigma(\mathcal{T}, \mathcal{M} \setminus \mathcal{B}) = \emptyset$ then
 8:             $\mathcal{M} := \mathcal{M} \setminus \mathcal{B}$
 9:         else
10:             if $|\mathcal{B}| > 1$ then
11:                 $(\mathcal{B}_l, \mathcal{B}_r) :=$ SPLITHALF($\mathcal{B}$)
12:                 $\mathbb{Q} := \mathcal{B}_l :: \mathcal{B}_r :: \mathbb{Q}$
13:             end if
14:         end if
15:     end while
16:     return $\mathcal{M}$
17: end function
```

Definition 8 (Logical Difference Core) *Let $\mathcal{T}$ be an $\mathcal{EL}$-terminology and let Σ be a signature. A subset $\mathcal{C} \subseteq \mathcal{T}$ is said to be a* logical difference core of $\mathcal{T}$ w.r.t. Σ *iff for every $\alpha \in \mathcal{C}$ it holds that $\mathsf{cDiff}_\Sigma(\mathcal{T}, \mathcal{T} \setminus \{\alpha\}) \neq \emptyset$.*

Given a logical difference core $\mathcal{C}$ of $\mathcal{T}$ w.r.t. Σ and a minimal module $\mathcal{M}$ of $\mathcal{T}$ w.r.t. Σ, it is easy to see that $\mathcal{C} \subseteq \mathcal{M}$ must hold. The maximal logical difference core can be computed by collecting all the axioms $\alpha \in \mathcal{T}$ for which $\mathsf{cDiff}_\Sigma(\mathcal{T}, \mathcal{T} \setminus \{\alpha\}) \neq \emptyset$.

Now, the procedure SINGLE-MINIMAL-MODULE-BUBBLE applied on a terminology $\mathcal{T}$, a signature Σ, an initial size parameter n for the bubbles, and a logical difference core $\mathcal{C}$ of $\mathcal{T}$ w.r.t. Σ operates as follows. First, the variable $\mathcal{M}$ is set to contain all the axioms of $\mathcal{T}$ and the bubble queue $\mathbb{Q}$ is initialised by partitioning the axioms contained in $\mathcal{T} \setminus \mathcal{C}$ into bubbles of size n. Note that the size of one bubble may be different from n if n is not a divisor of $|\mathcal{T}|$, or if $n > |\mathcal{T}|$. The resulting bubbles are then stored in the queue $\mathbb{Q}$. As long as $\mathbb{Q}$ is not empty, the first bubble $\mathcal{B}$ is extracted from the queue (lines 5 and 6). Note that the empty queue is denoted with $[]$. Subsequently, it is verified in Line 7 whether the removal of the axioms in $\mathcal{B}$ from the minimal module candidate $\mathcal{M}$ leads to a logical difference. If not, all the axioms in $\mathcal{B}$ can safely be removed from $\mathcal{M}$ in Line 8. Otherwise, if the bubble contained more than one axiom (Line 10), we have to identify the subsets of $\mathcal{B}$ whose removal does not yield a logical difference. To that end, $\mathcal{B}$ is split into two bubbles $\mathcal{B}_l$ and $\mathcal{B}_r$ (Line 11) such that $\mathcal{B}_l, \mathcal{B}_r \subseteq \mathcal{B}$, $|\mathcal{B}_l| = \lceil \frac{1}{2} \cdot |\mathcal{B}| \rceil$, and $|\mathcal{B}_r| = \lfloor \frac{1}{2} \cdot |\mathcal{B}| \rfloor$. The bubbles $\mathcal{B}_l$ and $\mathcal{B}_r$ are then prepended to the queue (Line 12), and the algorithm continues with the next iteration.

The correctness of Algorithm 2 can be shown as before with Algorithm 1. Termination on any input follows from the fact that every axiom in $\mathcal{T}$ appears in at most one bubble in $\mathbb{Q}$ and that in each iteration either the overall number of bubbles is reduced, or one bubble that contains more than one axiom is split into two smaller bubbles. Note that once a bubble $\mathscr{B}$ of size 1 has been selected in Line 5, it will not be contained in $\mathbb{Q}$ in subsequent iterations. We obtain the following result.

Theorem 9 *Let $\mathscr{T}$ be an $\mathscr{EL}$-terminology and let Σ be a signature. Additionally, let $\mathscr{C} \subseteq \mathscr{T}$ be a logical difference core of $\mathscr{T}$ w.r.t. Σ.*
Then Algorithm 2 applied on $\mathscr{T}$, Σ, and $\mathscr{C}$ computes a minimal module of $\mathscr{T}$ for Σ.

Regarding computational complexity, we observe that the decomposition of every bubble $\mathscr{B}$ induces a binary tree in which the nodes are labelled with the bubbles resulting from splitting the parent bubble. In our algorithm, given a bubble $\mathscr{B}$, such a decomposition tree has a depth of at most $\lfloor \log_2 |\mathscr{B}| \rfloor$ and the number of nodes in a decomposition tree corresponds to the number of logical difference checks. As the number of nodes in a binary tree of depth h is bounded by $2^{h+1} - 1$, we hence obtain that every initial bubble $\mathscr{B}$ results in at most $2 \cdot |\mathscr{B}| - 1$ logical difference checks. Overall, we can infer that the procedure SINGLE-MINIMAL-MODULE-BUBBLE runs in polynomial time in the size of $\mathscr{T}$, Σ, and n.

3.2. Computing All Minimal Modules

A naïve way to compute all minimal modules is to enumerate all subsets of the input TBox $\mathscr{T}$ and to check their logical difference w.r.t. $\mathscr{T}$ and a given signature. For $\mathscr{EL}$-terminologies the logical difference problem can be decided in polynomial time [6]. Example 6 shows that there are $\mathscr{EL}$-terminologies with exponentially many minimal modules. Consequently, computing all minimal modules of an $\mathscr{EL}$-terminology can only be achieved in time exponential in the size of the terminology in the worst case.

For that reason, upper approximations of (the union of) all minimal modules such as the syntactic locality-based module notions that can be extracted more efficiently have been introduced [4]. In our algorithm for computing all minimal modules (and in our experiments for extracting one minimal module) we will make use of syntactic $\bot\top^*$-locality modules to speed up computations. These modules are among the smallest modules based on syntactic locality notions [16]. They can be obtained by iterating the process of extracting a syntactic $\bot$-local module followed by extracting a syntactic $\top$-local module until a fixpoint is reached. We will extract syntactic $\bot\top^*$-locality modules using the OWLAPI.[2] Note that any syntactic $\bot\top^*$-locality module $\mathscr{M}_s$ of an $\mathscr{EL}$-terminology $\mathscr{T}$ w.r.t. a signature Σ contains all the minimal modules of $\mathscr{T}$ w.r.t. Σ.

In our algorithm for computing all minimal modules, we make use of a technique developed for computing all minimal hitting sets [15]. Our algorithm is based on the following observation: given a minimal module $\mathscr{M}$ of $\mathscr{T}$ w.r.t. a signature Σ, then for any other minimal module $\mathscr{M}'$ of $\mathscr{T}$ w.r.t. Σ there must exist $\alpha \in \mathscr{M}'$ such that $\alpha \notin \mathscr{M}$, i.e. $\mathscr{M}'$ must be contained in $\mathscr{T} \setminus \{\alpha\}$ for some $\alpha \in \mathscr{M}$.

Similarly to [15], our algorithm organises the search space using a labelled, directed tree τ, called *module search tree for $\mathscr{T}$*, that is extended during the run of the algorithm. Formally, τ is a tuple $(\mathscr{V}, \mathscr{E}, \mathscr{L}, \rho)$, where $\mathscr{V}$ is a non-empty, finite set of nodes, $\mathscr{E} \subseteq \mathscr{V} \times \mathscr{V}$ is a set of edges, $\mathscr{L}$ is an edge labelling function, mapping every edge $e \in \mathscr{E}$ to an axiom $\alpha \in \mathscr{T}$, and $\rho \in \mathscr{V}$ is the root node of τ. The procedure ALL-MINIMAL-MODULES shown in Algorithm 3 operates on a queue $\mathbb{Q}$ that contains

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

the nodes of τ that still have to be expanded. Intuitively, the labels of the edges on the unique path from the root node to a node $v \in \mathcal{V}$ are the axioms that should be excluded from the search for minimal modules. In each iteration a node v is extracted from $\mathbb{Q}$ and the set $\mathcal{T}_{\text{ex}} \subseteq \mathcal{T}$ of exclusion axioms is computed by analysing the path from the root node to v. The procedure SINGLE-MINIMAL-MODULE-BUBBLE is then used to find a minimal module $\mathcal{M}$ of $\mathcal{T} \setminus \mathcal{T}_{\text{ex}}$ w.r.t. Σ. Subsequently, the tree τ is extended by adding a child v_α of v for every $\alpha \in \mathcal{M}$ and the search for all minimal modules continues in the next iteration on the newly added nodes v_α.

Algorithm 3 Computing All Minimal Modules w.r.t. a Signature

INPUT: $\mathcal{EL}$-terminology $\mathcal{T}$, signature Σ, $n \geq 1$, logical difference core $\mathcal{C} \subseteq \mathcal{T}$ w.r.t. Σ

```
1: function ALL-MINIMAL-MODULES($\mathcal{T}, \Sigma, n, \mathcal{C}$)
2:     $\mathcal{T}_\Sigma := $ SYNTACTICLOCALITYMODULE($\mathcal{T}, \Sigma$)
3:     $\mathbb{M} := \emptyset$; $\tau := (\{\rho\}, \emptyset, \emptyset, \rho)$; $\mathbb{Q} := [\rho]$; $\mathbb{W} := \emptyset$
4:     while $\mathbb{Q} \neq []$ do
5:         $v := $ HEAD($\mathbb{Q}$)
6:         $\mathbb{Q} := $ TAIL($\mathbb{Q}$)
7:         $\mathbb{W} := \mathbb{W} \cup \{v\}$
8:         $\mathcal{T}_{\text{ex}} := $ LABELS(PATH($\tau, \rho, v$))
9:         if IS-PATH-REDUNDANT($\tau, \rho, \mathcal{T}_{\text{ex}}, \mathbb{W}$) then
10:            continue
11:        end if
12:        if $\mathsf{cDiff}_\Sigma(\mathcal{T}_\Sigma, \mathcal{T}_\Sigma \setminus \mathcal{T}_{\text{ex}}) \neq \emptyset$ then
13:            continue
14:        end if
15:        $\mathcal{M} := \emptyset$
16:        if there exists $\mathcal{M}' \in \mathbb{M}$ such that $\mathcal{T}_{\text{ex}} \cap \mathcal{M}' = \emptyset$ then
17:            $\mathcal{M} := \mathcal{M}'$
18:        else
19:            $\mathcal{M} := $ SINGLE-MINIMAL-MODULE-BUBBLE($\mathcal{T}_\Sigma \setminus \mathcal{T}_{\text{ex}}, \Sigma, n, \mathcal{C}$)
20:            if $\mathcal{M} = \mathcal{C}$ then
21:                return $\{\mathcal{C}\}$
22:            end if
23:            $\mathbb{M} := \mathbb{M} \cup \{\mathcal{M}\}$
24:        end if
25:        for every $\alpha \in \mathcal{M} \setminus \mathcal{C}$ do
26:            $v_\alpha := $ ADDCHILD($\tau, v, \alpha$)
27:            $\mathbb{Q} := v_\alpha :: \mathbb{Q}$
28:        end for
29:    end while
30:    return $\mathbb{M}$
31: end function
```

We now describe the ALL-MINIMAL-MODULES procedure in detail, together with the optimisations that we implemented. Some of the improvements to prune the search space have been proposed in [15] already.

Given an $\mathcal{EL}$-terminology $\mathcal{T}$, a signature Σ, a bubble size $n \geq 1$, and a logical difference core $\mathcal{C} \subseteq \mathcal{T}$ of $\mathcal{T}$ w.r.t. Σ as input, in the lines 2 and 3 a syntactic $\bot\top^*$-locality module $\mathcal{T}_\Sigma$ of $\mathcal{T}$ w.r.t. Σ is extracted from $\mathcal{T}$, the variable τ is initialised to represent a module search tree for $\mathcal{T}$ having only one node ρ. Moreover, the variables $\mathbb{M} \subseteq 2^{\mathcal{T}_\Sigma}$, containing the minimal modules that have been computed so far, and $\mathbb{W} \subseteq \mathcal{V}$, containing the already explored nodes of τ, are both initialised with the empty set. The queue $\mathbb{Q}$ of nodes in τ that still have to be explored is also set to contain the node ρ as its only element.

The algorithm then enters a while-loop in the lines 4 to 29 in which it remains as long as $\mathbb{Q}$ is not empty. In each iteration the first element v is extracted from $\mathbb{Q}$ and v is added to $\mathbb{W}$ (lines 5 to 7). Subsequently, the axioms labelling the edges of the path π_v from ρ to v in τ are collected in the set $\mathcal{T}_{\text{ex}}$ (Line 8). The algorithm then checks whether π_v is redundant, in which case the next iteration of the while-loop starts.

The path π_v is redundant iff there exists an already explored node $w \in \mathbb{W}$ such that (a) the axioms in $\mathcal{T}_{\text{ex}}$ are exactly the axioms labelling the edges of the path π_w from ρ to w in τ, or (b) w is a leaf node of τ and the edges of π_w are only labelled with axioms from $\mathcal{T}_{\text{ex}}$. Condition (a) corresponds to *early path termination* in [5, 15]: the existence of π_w implies that all possible extensions of π_v have already been considered. Condition (b) implies that the axioms labelling the edges of π_w lead to a logical difference when removed from $\mathcal{T}_\Sigma$. Consequently, removing $\mathcal{T}_{\text{ex}}$ from $\mathcal{T}_\Sigma$ also induces a logical difference by monotonicity of $\models$, implying that π_v and all its extensions do not have to be explored. Moreover, the current iteration can also be terminated immediately if $\mathsf{cDiff}_\Sigma(\mathcal{T}_\Sigma, \mathcal{T}_\Sigma \setminus \mathcal{T}_{\text{ex}}) \neq \emptyset$ (lines 12 to 14) as no subset of $\mathcal{T}_\Sigma \setminus \mathcal{T}_{\text{ex}}$ can be a module of $\mathcal{T}_\Sigma$ (and therefore of $\mathcal{T}$) w.r.t. Σ.

Subsequently, in Line 15 the variable $\mathcal{M}$ that will hold a minimal module of $\mathcal{T}_\Sigma \setminus \mathcal{T}_{\text{ex}}$ is initialised with $\emptyset$. At this point we can check if a minimal module $\mathcal{M}' \in \mathbb{M}$ has already been computed for which $\mathcal{T}_{\text{ex}} \cap \mathcal{M}' = \emptyset$ (lines 16 and 17) holds, in which case we set $\mathcal{M}$ to $\mathcal{M}'$. This optimisation step can also be found in [5,15] and it allows us to avoid a costly call to the SINGLE-MINIMAL-MODULE-BUBBLE procedure. Otherwise, in the lines 18 to 24 we have to apply SINGLE-MINIMAL-MODULE-BUBBLE on $\mathcal{T}_\Sigma \setminus \mathcal{T}_{\text{ex}}$ to obtain a minimal module of $\mathcal{T}_\Sigma \setminus \mathcal{T}_{\text{ex}}$ w.r.t. Σ. The algorithm then checks whether $\mathcal{M}$ is equal to $\mathcal{C}$ (lines 20 to 22), in which case the search for additional modules can be aborted. If the logical difference core $\mathcal{C}$ is a minimal module itself, we can infer that no other minimal module exists since $\mathcal{C}$ is a subset of all the minimal modules. Otherwise, the module $\mathcal{M}$ is added to $\mathbb{M}$ in Line 23. Finally, in the lines 25 to 28 the tree τ is extended by adding a child v_α to v for every $\alpha \in \mathcal{M} \setminus \mathcal{C}$, connected by an edge labelled with α. Note that it is sufficient to take $\alpha \notin \mathcal{C}$ as a set $\mathcal{M}$ with $\mathcal{C} \nsubseteq \mathcal{M}$ cannot be a minimal module of $\mathcal{T}$ w.r.t. Σ. The procedure finishes by returning the set $\mathbb{M}$ in Line 30.

Regarding correctness of Algorithm 3, we note that only minimal modules are added to $\mathbb{M}$. For completeness, one can show that the locality-based module $\mathcal{T}_\Sigma$ of $\mathcal{T}$ w.r.t. Σ contains all the minimal modules of $\mathcal{T}$ w.r.t. Σ. Moreover, it is easy to see that the proposed optimisations do not lead to a minimal module not being computed. Overall, we obtain the following result.

$\lvert\Sigma\cap\mathsf{N_C}\rvert$	50			
Bubble Size	10	25	50	100
Time (s)	67 / 523 / 200 / 87.2	68 / 483 / 197 / 82.5	70 / 505 / 202 / 85.3	71 / 507 / 204 / 86.4
Sizes	50 / 118 / 77 / 14.6	50 / 118 / 77 / 14.5	50 / 118 / 77 / 14.6	50 / 118 / 77 / 14.6
Size MEX-Mod	401 / 720 / 587 / 60.7			
Size $\bot\top^*$-Mod	1075 / 2456 / 1803 / 300.2			
$\lvert\Sigma\cap\mathsf{N_C}\rvert$	75			
Bubble Size	10	25	50	100
Time (s)	225 / 584 / 434 / 102.0	216 / 1359 / 531 / 209.8	226 / 575 / 447 / 101.0	231 / 586 / 450 / 101.8
Sizes	78 / 177 / 110 / 16.4	75 / 216 / 113 / 20.8	78 / 177 / 110 / 15.8	78 / 178 / 110 / 15.9
Size MEX-Mod	679 / 971 / 825 / 72.0			
Size $\bot\top^*$-Mod	1939 / 3779 / 2641 / 373.2			

Table 1. Computation of *one* minimal basic $\mathcal{EL}$-subsumption module of Snomed CT for 100 random signatures containing 50/75 concept names and all role names (minimal / maximal / median / standard deviation)

Theorem 10 *Let $\mathcal{T}$ be an $\mathcal{EL}$-terminology and let Σ be a signature. Additionally, let $n \geq 1$, and let $\mathcal{C} \subseteq \mathcal{T}$ be a logical difference core of $\mathcal{T}$ w.r.t. Σ.*

Then the procedure ALL-MINIMAL-MODULES *shown in Algorithm 3 and applied on $\mathcal{T}$, Σ, n, and $\mathcal{C}$, exactly computes all the minimal modules of $\mathcal{T}$ for Σ.*

Algorithm 3 terminates on any input as the paths in the module search tree τ for $\mathcal{T}$ that is constructed during the execution represent all the permutations of the axioms in $\mathcal{T}$ that are relevant for finding all minimal modules. It is easy to see that the procedure ALL-MINIMAL-MODULES runs in exponential time in size of $\mathcal{T}$ (and polynomially in Σ, n, and $\mathcal{C}$) in the worst case.

4. Evaluation

To demonstrate the practical applicability of our approach, we have implemented Algorithms 2 and 3 in a Java prototype to compute one and all minimal basic $\mathcal{EL}$-subsumption modules of $\mathcal{EL}$-versions (i.e., without role axioms) of two prominent biomedical ontologies: Snomed CT (version Jan 2016), an acyclic $\mathcal{EL}$-terminology consisting of 317 891 axioms, and NCI (version 14.01d), a cyclic $\mathcal{EL}$-terminology containing 105 280 axioms. The experiments have been carried out on machines equipped with an Intel Xeon Core 4 Duo CPU running at 2.50GHz and with 64GiB of RAM.

Tables 1 and 2 show the results for computing one minimal basic $\mathcal{EL}$-subsumption module of Snomed CT and NCI for 100 random signatures of different sizes. When the size of the signature increases, it takes more time in general to compute one minimal module and the size of their minimal module is also increasing. Moreover, in our experiments the median computation times were decreasing with an increasing bubble size for signatures with 200 concept names.

Table 3 shows that there exist several minimal basic $\mathcal{EL}$-subsumption modules of Snomed CT for the selected signatures (which contain concept names connected to at most 8 other axioms). Note that we did not consider signatures that are extracted at random as they usually yield one minimal module only. In our experiments the number of minimal modules rose up to 32, and the size of the minimal modules varied from one signature to another.

$\lvert\Sigma \cap \mathsf{N_C}\rvert$	100			
Bubble Size	10	50	100	200
Time (s)	5 / 258 / 116 / 42.3	2 / 84 / 15 / 14.8	1 / 89 / 2 / 22.2	1 / 86 / 9 / 15.7
Sizes	0 / 17 / 0 / 3.1	0 / 17 / 0 / 3.1	0 / 64 / 0 / 17.3	0 / 17 / 0 / 3.1
Size $\bot\top^*$-Mod	679 / 3895 / 3510 / 915.2			
$\lvert\Sigma \cap \mathsf{N_C}\rvert$	200			
Bubble Size	10	50	100	200
Time (s)	125 / 581 / 210 / 100.7	28 / 546 / 59 / 99.4	16 / 552 / 39 / 105.3	9 / 562 / 29 / 109.5
Sizes	0 / 103 / 3 / 18.4	0 / 103 / 3 / 19.7	0 / 103 / 3 / 19.7	0 / 103 / 3 / 19.7
Size $\bot\top^*$-Mod	3670 / 4619 / 4003 / 196.7			

Table 2. Computation of *one* minimal basic $\mathcal{EL}$-subsumption module of NCI for 100 random signatures containing 100/200 concept names and all role names (minimal / maximal / median / standard deviation)

Optimisation	Core $\mathscr{C} = \emptyset$	Core $\mathscr{C} \neq \emptyset$
Time (s)	5 / 709 / 24 / 238.7	2 / 118 / 7 / 36.0
Number of Modules	1 / 32 / 6 / 9.2	
Size of Modules	81 / 265 / 126 / 46.5	

Table 3. Computation of *all* minimal basic $\mathcal{EL}$-subsumption modules of Snomed CT for 20 selected signatures consisting of 30 concept names and all role names using a bubble size of 50 (min / max / med / std dev)

Although a precomputation of the maximal logical difference core has the potential of narrowing down the search space, it requires extra computational effort, which can be potentially very time-consuming. In order to check whether the use of the logical difference core can help to speed up the process of searching for all minimal modules, we computed all the minimal modules of Snomed CT with and without precomputing the maximal logical difference core for the same signatures. It turns out that in our experiments the precomputation of the maximal core was beneficial to the overall performance: the overall computation process was sped up by more than three times.

5. Conclusion

We have reused the black-box approach for computing justifications in order to devise two algorithms for computing one or all basic $\mathcal{EL}$-subsumption modules. We deploy a version of CEX as an oracle for determining whether two possibly cyclic $\mathcal{EL}$-terminologies are logically different (i.e. not inseparable). Our algorithms are applicable to ontologies formulated in any ontology language provided that a tool is available that can effectively decide the inseparability notion of interest.

Our algorithms may require many costly calls to a logical difference tool. One way to reduce the overall computation time would be to use a tool that allows for an iterative computation of the logical difference (i.e., a tool that utilises previous computations on similar input to determine the existence of a logical difference faster). Another possible optimisation is refining the single module search algorithm by deploying a strategy for selecting the sets of axioms (bubbles) that are to be removed next from the minimal module candidate. Moreover, when creating bubbles (Algorithm 2) or selecting axioms that are to be excluded from minimal modules (Algorithm 3) one can ensure that axioms that always co-occur in minimal basic $\mathcal{EL}$-subsumption modules are not separated. Finally, instead of searching for minimal modules in the entire ontology, our algorithm first extracts modules that are based on the notion of syntactic locality. A further optimisation might be achieved by exploring ways to compute such modules faster [14].

Acknowledgements. This work was partially supported by the German Research Foundation (DFG) within the Cluster of Excellence 'Center for Advancing Electronics Dresden' and the China Scholarships Council. We would also like to thank the reviewers of the workshop *WOMoCoE 2016* and Yue Ma (Laboratoire de Recherche en Informatique, Université Paris-Sud, France) for helpful feedback and input.

References

[1] F. Baader, S. Brandt, and C. Lutz. Pushing the EL envelope further. In *In Proceedings of the OWLED 2008 DC Workshop on OWL: Experiences and Directions*, 2008.

[2] F. Baader, R. Peñaloza, and B. Suntisrivaraporn. Pinpointing in the description logic EL. In *Proceedings of KI'07*, volume 4667 of *LNAI*, pages 52–67, Osnabrück, Germany, 2007. Springer-Verlag.

[3] A. Ecke, M. Ludwig, and D. Walther. The concept difference for EL-terminologies using hypergraphs. In *Proceedings of the International workshop on (Document) Changes: modeling, detection, storage and visualization (DChanges 2013)*, volume 1008 of *CEUR-WS*, 2013.

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

[5] A. Kalyanpur, B. Parsia, M. Horridge, and E. Sirin. Finding all justifications of OWL DL entailments. In *Proceedings of the 6th International Semantic Web Conference & 2nd Asian Semantic Web Conference (ISWC 2007 & ASWC 2007)*, volume 4825 of *LNCS*, pages 267–280. Springer, 2007.

[6] B. Konev, M. Ludwig, D. Walther, and F. Wolter. The logical difference for the lightweight description logic EL. *Journal of Artificial Intelligence Research (JAIR)*, 44:633–708, 2012.

[7] B. Konev, M. Ludwig, and F. Wolter. Logical difference computation with CEX2.5. In *Proceedings of IJCAR'12*, pages 371–377, Berlin, Heidelberg, 2012. Springer-Verlag.

[8] B. Konev, C. Lutz, D. Walther, and F. Wolter. Semantic modularity and module extraction in description logics. In *Proceedings of ECAI'08*, pages 55–59, Amsterdam, The Netherlands, 2008. IOS Press.

[9] B. Konev, C. Lutz, D. Walther, and F. Wolter. Model-theoretic inseparability and modularity of description logic ontologies. *Artificial Intelligence*, 203:66–103, 2013.

[10] B. Konev, D. Walther, and F. Wolter. The logical difference problem for description logic terminologies. In *Proceedings of IJCAR'08*, pages 259–274, Berlin, Heidelberg, 2008. Springer-Verlag.

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

[12] M. Ludwig and D. Walther. The logical difference for ELHr-terminologies using hypergraphs. In *Proceedings of ECAI'14*, volume 263 of *Frontiers in Artificial Intelligence and Applications*, pages 555–560. IOS Press, 2014.

[13] C. Lutz and F. Wolter. Deciding inseparability and conservative extensions in the description logic EL. *Journal of Symbolic Computation*, 45(2):194–228, Feb. 2010.

[14] F. Martin-Recuerda and D. Walther. Fast modularisation and atomic decomposition of ontologies using axiom dependency hypergraphs. In *Proceedings of ISWC'14, Part II*, volume 8797 of *LNCS*, pages 49–64. Springer-Verlag, 2014.

[15] R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95, 1987.

[16] U. Sattler, T. Schneider, and M. Zakharyaschev. Which kind of module should I extract? In *Proceedings of DL'09*, volume 477 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2009.