CEX and MEX: Logical Diff and Semantic Module Extraction in a Fragment of OWL

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

The ontology language OWL, a W3C recommendation, is currently being revisited by a W3C working group. It is anticipated that the updated version of the W3C recommendation produced by the working group will include a number of popular fragments of OWL that have more favourable computational properties than the full language. One candidate for such a fragment is an extension of the lightweight description logic $\mathcal{EL}$, which only provides conjunction and existential restriction. It is well-known that, in $\mathcal{EL}$ and many of its extensions, satisfiability, subsumption, and other standard reasoning problems can be decided in polynomial time. In this paper, we give further evidence for the attractive computational properties of $\mathcal{EL}$ by analyzing the practical feasibility of two more involved reasoning problems, which can also be solved in polynomial time: logical diff and the extraction of semantic modules.

The standard diff operation for text files is an indispensible tool for comparing different versions of text files and source code files. In contrast, a purely syntactic diff operation is hardly useful to compare ontologies, see e.g. [10]. Indeed, one is usually not interested in syntactic differences between ontologies, but rather in different *consequences* that the ontologies have. We use a logic-based diff operation that compares the consequences of two ontologies, and is closely related to the notion of a (deductive) conservative extension as studied in [8, 9, 5].

The purpose of module extraction is to identify, given a signature Σ and an ontology $\mathcal{T}$, a (preferably small) fragment $\mathcal{T}_0$ of $\mathcal{T}$ such that $\mathcal{T}_0$ contains the same information about Σ as $\mathcal{T}$ and thus behaves in exactly the same way as $\mathcal{T}$ in all applications using only symbols from Σ. Possible applications include (a) importing, instead of $\mathcal{T}$, the ontology $\mathcal{T}_0$ into another ontology, (b) computing the classification of the terms in Σ, and (c) querying a database using $\mathcal{T}_0$ instead of $\mathcal{T}$. We use a logic-based, semantic definition of a module based on (model-theoretic) conservative extensions which ensures that $\mathcal{T}$ and $\mathcal{T}_0$ "contain the same information about Σ" in a very strong sense.

This paper is structured into two parts. The first part presents experiments with the system CEX. This prototype computes, given two ontologies formulated as (cyclic or acyclic) TBoxes in the description logic $\mathcal{EL}$ and a signature Σ, a complete list of concept names that are involved in a subsumption between (possibly complex) Σ-concepts that is a consequence of one TBox, but not of the other.

By defining Σ as the set of shared symbols of the two TBoxes, we thus obtain a list that can be regarded as the logical difference. Most of the experiments with CEX are based on (different versions of) Snomed ct, the Systematized Nomenclature of Medicine, Clinical Terms. This acyclic TBox comprises $\sim$0.4 million terms and underlies the systematized medical terminology used in the health systems of the US, the UK, and other countries [12].

The second part of this paper presents experiments with the system MEX that extracts modules from ontologies formulated as acyclic $\mathcal{EL}$-TBoxes. Similar to the systems described in [5, 2, 4, 11, 3], MEX takes as input a TBox $\mathcal{T}$ and a set of symbols Σ, and extracts a self-contained module $\mathcal{T}_0$ of $\mathcal{T}$ that contains the same information about Σ as $\mathcal{T}$ and is minimal with this property in a certain sense. In contrast to most existing module extraction algorihms, MEX does not collect the definitions of terms in Σ, but computes exactly what is required in the applications mentioned above. The experiments for MEX are also based on Snomed ct. Proofs and additional results are available at [7, 6].

2 Preliminaries

Let $\mathsf{N_C}$ and $\mathsf{N_R}$ be countably infinite and disjoint sets of *concept names* and *role names*, respectively. In the description logic $\mathcal{EL}$, *concepts* C are built according to the syntax rule

$$C ::= \top \mid A \mid C \sqcap D \mid \exists r.C,$$

where A ranges over $\mathsf{N_C}$, r ranges over $\mathsf{N_R}$, and C, D range over concepts. 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 extended to arbitrary concepts by setting $\top^{\mathcal{I}} := \Delta^{\mathcal{I}}$, $(C \sqcap D)^{\mathcal{I}} := C^{\mathcal{I}} \cap D^{\mathcal{I}}$, and $(\exists r.C)^{\mathcal{I}} := \{d \in \Delta^{\mathcal{I}} \mid \exists e \in C^{\mathcal{I}} : (d, e) \in r^{\mathcal{I}}\}$.

A TBox $\mathcal{T}$ is a finite set of *axioms*, where an axiom can be a *concept inclusion (CI)* of the form $A \sqsubseteq C$ or a *concept equation (CE)* of the form $A \equiv C$ with A a concept name. It is required that no concept name occurs more than once on the left hand side of an axiom in $\mathcal{T}$. Define the relation $\prec_{\mathcal{T}} \subseteq \mathsf{N_C} \times \mathsf{N_C}$ 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 concept names which are used in the definition of A in $\mathcal{T}$. A TBox $\mathcal{T}$ is called *acyclic* if $A \notin \mathsf{depend}_{\mathcal{T}}(A)$ for any $A \in \mathsf{N_C}$.

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 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 $C \sqsubseteq D$ ($C \equiv D$, respectively).

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.

3 Logical Difference

In this section, we present the experimental results for CEX. We first give a logical characterization of the algorithm underlying CEX.

Description of CEX : For any two $\mathcal{EL}$-TBoxes $\mathcal{T}_0$ and $\mathcal{T}_1$ and signature Σ, the algorithm implemented in CEX outputs, in polytime, the following two lists:

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

One can show that, if there exist Σ-concepts C and D such that $\mathcal{T}_1 \models C \sqsubseteq D$ but $\mathcal{T}_0 \not\models C \sqsubseteq D$, then an $A \in \Sigma$ occuring in C or D is in $\mathsf{DiffR}_\Sigma(\mathcal{T}_0, \mathcal{T}_1)$ or $\mathsf{DiffL}_\Sigma(\mathcal{T}_0, \mathcal{T}_1)$, respectively. In particular, both lists are empty if, and only if, $\mathcal{T}_0$ and $\mathcal{T}_1$ do not differ w.r.t. Σ. We illustrate the definition using the following example. Let $\mathcal{T}_0$ be the TBox containing the following three axioms:

$$\mathsf{Neck\_injection} \sqsubseteq \mathsf{Operation} \tag{1}$$

$$\mathsf{Neck\_operation} \sqsubseteq \mathsf{Operation} \tag{2}$$

$$\mathsf{Removal\_foreign\_body\_from\_neck} \equiv \mathsf{Neck\_operation} \sqcap \mathsf{Removal\_foreign\_body} \tag{3}$$

Assume that $\mathcal{T}_0$ is refined by replacing axiom (1) by the axiom $\mathsf{Neck\_injection} \sqsubseteq \mathsf{Neck\_operation}$. Let $\mathcal{T}_1$ be the new TBox. To find out how this update influences the relationship between concepts distinct from Neck_operation, one computes the difference between $\mathcal{T}_0$ and $\mathcal{T}_1$ for the signature Σ consisting of all symbols distinct from Neck_operation. Then $\mathsf{Removal\_foreign\_body\_from\_neck} \in \mathsf{DiffR}_\Sigma(\mathcal{T}_0, \mathcal{T}_1)$ because of $\mathcal{T}_1$ implying $\mathsf{Neck\_injection} \sqcap \mathsf{Removal\_foreign\_body} \sqsubseteq \mathsf{Removal\_foreign\_body\_from\_neck}$. Notice that no difference between $\mathcal{T}_0$ and $\mathcal{T}_1$ is visible by comparing the induced class hierarchies over Σ.

CEX is an OCaml program [13]. The experiments 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 $\mathcal{EL}$-TBoxes only, the role inclusions of SNOMED CT are not taken into account. The tests have been carried out on a standard PC: Intel® Core$^{\text{TM}}$ 2 CPU at 2.13 GHz and 3 GB of RAM.

Logical difference between SM-05 *and* SM-06*.* Table 1 shows the average time and memory consumption of CEX computing the lists $\mathsf{DiffR}_\Sigma(\text{SM-05}, \text{SM-06})$ and $\mathsf{DiffL}_\Sigma(\text{SM-05}, \text{SM-06})$ and vice versa for 20 randomly generated signatures Σ of size 100, 1 000, etc. The average size of the sets $\mathsf{DiffR}_\Sigma(\text{SM-05}, \text{SM-06})$ and $\mathsf{DiffL}_\Sigma(\text{SM-05}, \text{SM-06})$ are provided. Observe that no differences have been

Size of Σ	CEX: Diff(SM-05,SM-06)				CEX: Diff(SM-06,SM-05)			
	Time (Sec.)	Memory (MByte)	$\vert\mathsf{DiffL}_\Sigma\vert$	$\vert\mathsf{DiffR}_\Sigma\vert$	Time (Sec.)	Memory (MByte)	$\vert\mathsf{DiffL}_\Sigma\vert$	$\vert\mathsf{DiffR}_\Sigma\vert$
100	513.1	1 393.7	0.0	0.0	514.9	1 393.5	0.0	0.0
1 000	512.4	1 394.6	2.5	2.5	514.7	1 395.2	1.5	2.5
10 000	517.7	1 424.3	183.2	122.0	519.9	1 424.7	194.4	123.3
100 000	559.8	1 473.2	11 322.1	4 108.5	563.1	1 472.6	11 869.7	4 119.4

Table 1. Logical difference between two SNOMED CT versions

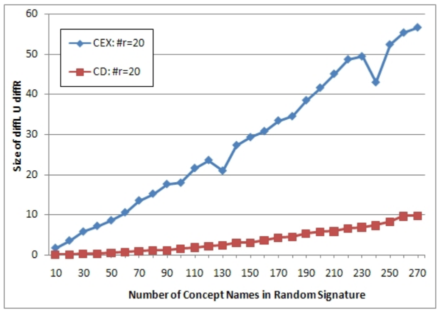
found for signatures of size 100. This means that the two versions of SNOMED CT are not distinguishable by any implied subsumptions formulated with concept names from the 20 randomly generated signatures of size 100.

Comparison with the classification approach. We compare the size of $\mathsf{DiffL}_\Sigma \cup \mathsf{DiffR}_\Sigma$ as computed by CEX with the number of concept names $A \in \Sigma$ for which there is a difference in the class hierarchy restricted to Σ. The experiments show how many of the differences between two TBoxes detected by CEX can be extracted from a straightforward comparison of class hierarchies.

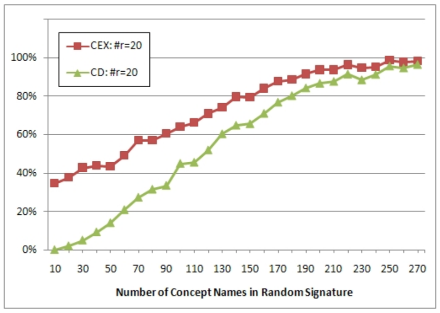
To facilitate the experiments, we use an empty TBox and an SM-05 fragment containing about 140 000 axioms. For every number between 10 and 270 with the step of 10, we generated 500 samples of a random signature containing this number of concepts and 20 roles. The results of the experiments are given in Figure 1. (a) shows that, for these signatures, the number of concept names CEX outputs is about five times larger than the number of concept names occuring in differences between the class hierarchies. In (b), we do not count the number of differences but analyse how often the two approaches detect differences at all. More precisely, we give the percentage of cases when CEX detects a difference between the two TBoxes and when a difference is visible in the class hierarchies. For signatures larger than 200, both approaches almost always detect differences. But for smaller signatures there is again a significant gap between the two approaches.

We note that the gap between differences detected by CEX and differences visible in class hierarchies is less significant if less roles names are in the signature Σ. But experiments show that even for signatures without role names CEX often detects differences that do not occur in the class hierarchy.

Scalability. We demonstrated in the previous section that CEX is capable of finding the logical difference in two unmodified versions of SNOMED CT. In order to see how CEX's performance scales, we now test it on randomly generated acyclic TBoxes of various sizes. Each randomly generated TBox contains a certain number of defined- and primitive concept names and role names. The ratio between concept equations and concept inclusions is fixed, as is the ratio between existential restrictions and conjunctions. The random TBoxes were generated for a

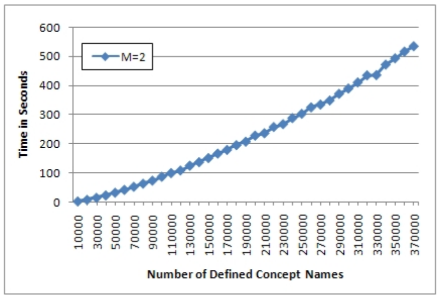


(a) Difference size

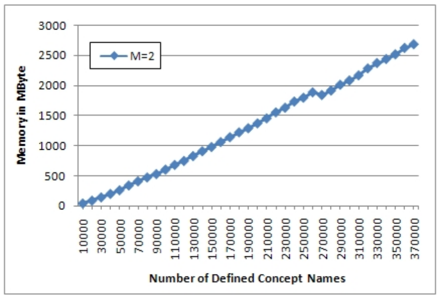


(b) Proportion

Fig. 1. Comparison of CEX and classification-based approach



(a) Execution time



(b) Memory consumption

Fig. 2. Performance of CEX on randomly generated TBoxes

varying number of defined concept names using the parameters of SM-05: 62 role names; the average number of conjuncts is 2.59; the equality-inclusion ratio is 0.102; and the exists-conjunction ratio is 0.652. For every chosen size, we generate a number of samples consisting of two random TBoxes as described above. We apply CEX to find the logical difference of the two TBoxes over their joint signature. Figure 2 shows the time and memory consumption of CEX on randomly generated TBoxes of various sizes, where the maximum length of conjunctions was fixed as two (M=2).

The performance of CEX crucially depends on the length of conjunctions as illustrated in Figures 3 and 4, where the number of conjuncts in each conjunction is randomly selected between two and M. The curves break off at the point where CEX runs out of memory. For example, in the case $M = 22$ this happens for TBoxes with more than 9 500 defined concept names.

4 Semantic module extraction

The purpose of our second tool, MEX, is to extract modules from a TBox. We use the following, logic-based definition of a module.

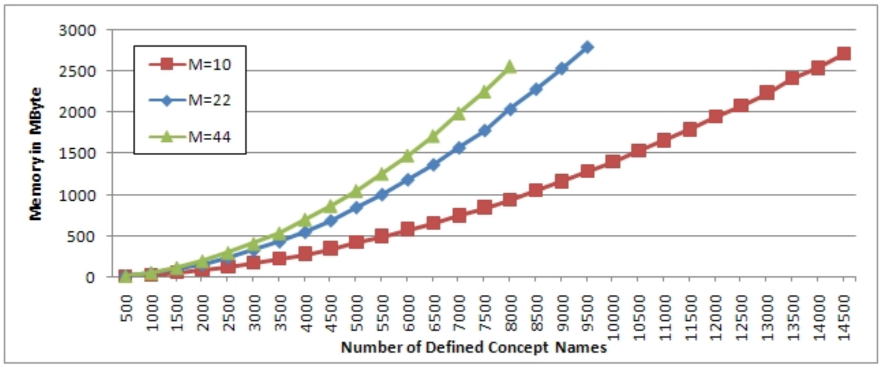


Fig. 3. CEX's memory consumption for TBoxes with long conjunctions

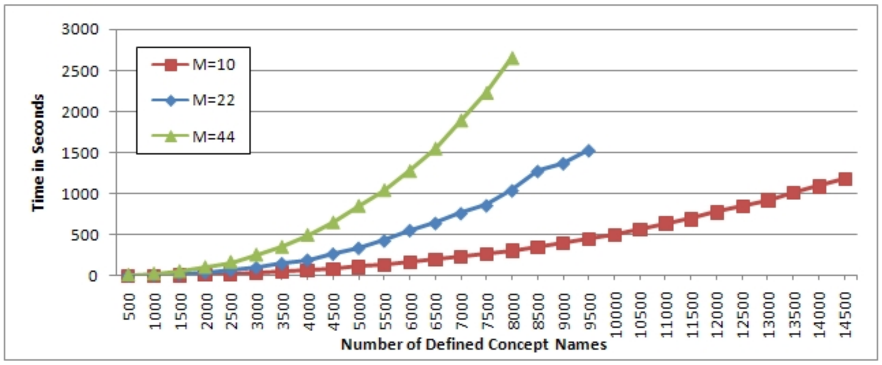


Fig. 4. CEX's execution time on TBoxes with long conjunctions

Definition 1 (Semantic modules). *Let $\mathcal{T}_0 \subseteq \mathcal{T}_1$ be TBoxes and $\Sigma \supseteq \mathsf{sig}(\mathcal{T}_0)$.*

- *$\mathcal{T}_0$ is a* weak semantic Σ-module of $\mathcal{T}_1$ *if for every model $\mathcal{I}$ of $\mathcal{T}_0$ there exists a model $\mathcal{I}'$ of $\mathcal{T}_1$ which coincides with $\mathcal{I}$ on Σ.*
- *$\mathcal{T}_0$ is a* strong semantic Σ-module of $\mathcal{T}_1$ *if for every model $\mathcal{I}$ there exists a model $\mathcal{I}'$ of $\mathcal{T}_1 \setminus \mathcal{T}_0$ which coincides with $\mathcal{I}$ on Σ.*

In any standard description logic, every strong semantic Σ-module is a weak semantic Σ-module. The converse does not hold: if $\mathcal{T}_0 = \{A \equiv \top\}$, $\mathcal{T}_1 = \mathcal{T}_0 \cup \{B \sqsubseteq A\}$ and $\Sigma = \{A, B\}$, then $\mathcal{T}_0$ is a weak semantic Σ-module of $\mathcal{T}_1$, but not a strong semantic Σ-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. This stronger type of module has been introduced in [5].

Call an axioms α *trivial* if it is of the form $A \equiv \top$, or $A \equiv \top \sqcap \top$, etc. Surprisingly, for acyclic $\mathcal{EL}$-TBoxes not containing trivial axioms the notions of weak and strong semantic modules are equivalent:

Theorem 1. *Let $\mathcal{T}_0 \subseteq \mathcal{T}_1$ be acyclic $\mathcal{EL}$-TBoxes not containing trivial axioms and $\Sigma \supseteq \mathsf{sig}(\mathcal{T}_0)$. Then $\mathcal{T}_0$ is a weak semantic Σ-module of $\mathcal{T}_1$ iff it is a strong semantic Σ-module of $\mathcal{T}_1$.*

We now discuss our tool MEX for extracting semantic modules.

Description of MEX. For any acyclic $\mathcal{EL}$-TBox $\mathcal{T}_1$ not containing trivial axioms and signature Σ, the algorithm implemented in MEX extracts the (uniquely determined) smallest weak/strong semantic $\Sigma \cup \mathsf{sig}(\mathcal{T}_0)$-module $\mathcal{T}_0$ of $\mathcal{T}_1$.

Given Σ, the algorithm may thus return a Σ'-module for some $\Sigma' \supseteq \Sigma$. Intuitively, the purpose is to make the module self-contained in the sense that, if a TBox $\mathcal{T}$ implies a dependency between symbols occuring in $\mathcal{T}_0$, then this dependency is implied by $\mathcal{T}_0$ already.

The size of modules. We compare the modules generated by MEX with the minimal modules generated by a number of other extraction algorithms. It is not difficult to see that, when applied to an acyclic $\mathcal{EL}$-TBox $\mathcal{T}_1$ and signature Σ, the module extraction algorithms presented in [5, 4, 11] 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 [5] (whereas the $\bot$-based locality approach yields definition-closed modules). One can show that any definition-closed module contains the module generated by MEX.

Definition-closed modules are appropriate for several applications of extraction algorithms. For applications that do not need definitions of terms in the input-signature, however, the smallest weak/strong semantic module appears to provide exactly the information required. We show that this can significantly reduce the size of modules. The following experiment compares the minimal size of definition-closed modules with the size of modules generated using MEX, when applied to SNOMED CT. We note that the system MEX takes into account also the role inclusions of SNOMED CT. In the experiments below, it outputs a weak (equivalently, strong) semantic module of SNOMED CT, but because of the role box this output of MEX is not necessarily a minimal semantic module. To compute the minimal definition-closed modules, we use the module extraction feature of the CEL reasoner [1] (Version 1.0b).

In Figure 5, 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. For every signature size, we use 1000 random signatures.

Figure 5 shows the maximal, minimal, and average module sizes depending on the size of the input signature. Figure 6 shows the frequency distribution of the definition-closed modules, and Figure 7 the distribution for the semantic modules. In each figure, there are five different histograms, one for each of the signature sizes ranging over 100, 250, 500, 750, and 1 000. Each of these histograms displays the distribution of the module sizes of 1 000 extracted SM-05 modules for randomly selected signatures of a certain size. For instance, the histogram labelled with CEL100 in Figure 6 shows the distribution of the size of

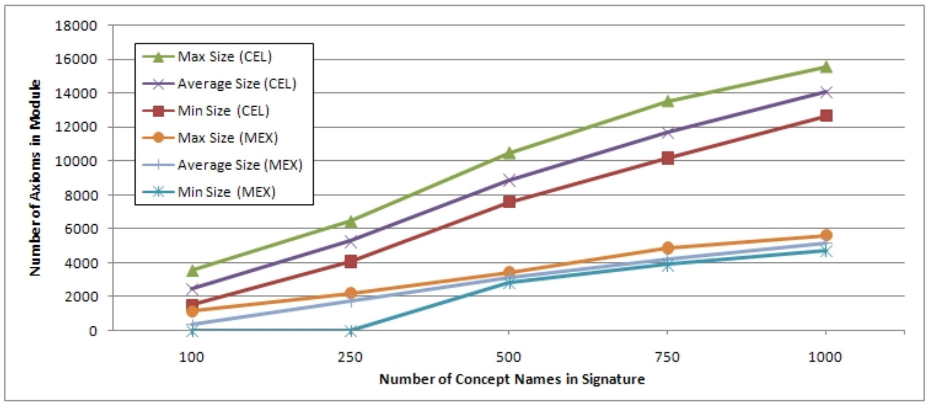


Fig. 5. Sizes of definition-closed modules and semantic modules

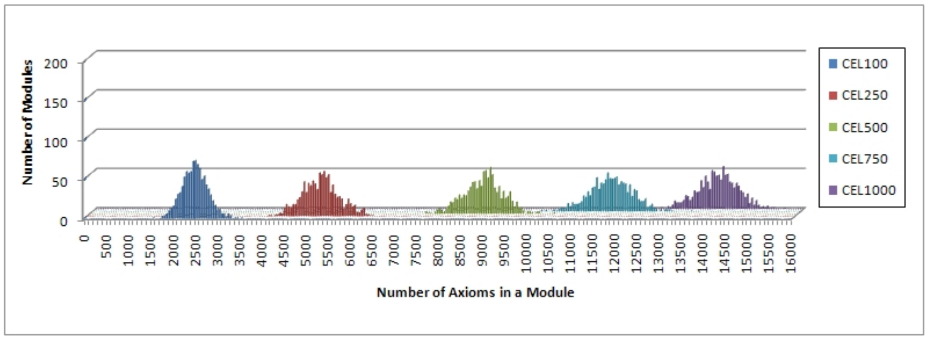


Fig. 6. Frequency distribution of the size of locality-based modules

1 000 definition-closed modules for the signature size 100 extracted from SM-05. For the sake of comparison, the axes in both figures have the same scaling resulting in the histogram MEX100 being capped at 200 for empty semantic modules. The missing value of MEX100 for empty modules is 547.

To facilitate the comparison of the module sizes, consider Table 2. It presents the average module size together with the standard deviation of definition-closed and semantic modules for random input signatures of various sizes. Recall that the standard deviation indicates how much the module sizes vary from the average. Notice that, for small signature sizes, the standard deviation of semantic module sizes is relatively high. The reason is that MEX extracts many small or even empty semantic modules for small signature sizes. For instance, 547 of 1 000 extracted modules where empty for signature size 100. Intuitively, the reason for an empty module is that SNOMED CT does not imply any subsumptions between concepts formulated in the chosen signature. When only considering the semantic modules for signature size 100 that contain more than 10 axioms, the average module size becomes 889.15 and the standard deviation decreases to 125.63; see the last column of Table 2.

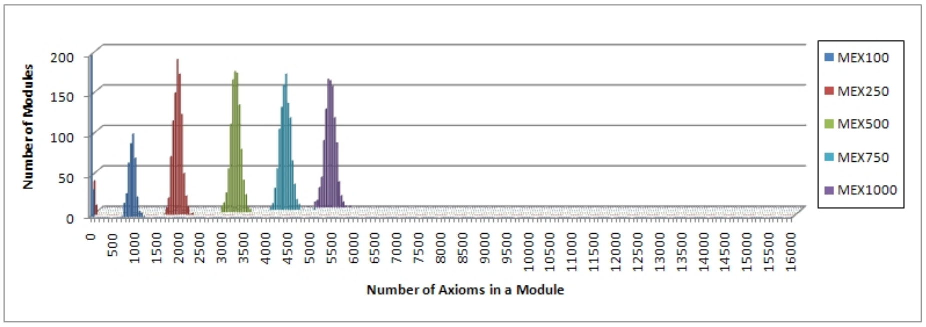


Fig. 7. Frequency distribution of the size of semantic modules

Signature size	Size of definition-closed modules		Size of semantic modules		Size of semantic modules of Size > 10	
	Average	Standard deviation	Average	Standard deviation	Average	Standard deviation
100	2 462.17	293.49	370.10	447.08	889.15	125.63
250	5 253.21	419.08	1 774.53	434.66	1 875.80	98.04
500	8 872.74	441.92	3 138.25	110.84	3 138.25	110.84
750	11 691.71	478.83	4 210.94	121.40	4 210.94	121.40
1 000	14 053.48	462.09	5 167.07	122.76	5 167.07	122.76

Table 2. Average and std. deviation definition-closed and semantic modules

5 Combining CEX and MEX

In this section, we show that MEX can be used to speed-up the computation of the logical difference between TBoxes. The left hand side of Table 3 is taken from Table 1 while its right hand side 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 for small Σ.

6 Discussion

In this paper, we have proposed the novel notion of a logical diff and presented a new logic-based notion of a module in a TBox. In both cases, we have developed polytime algorithms and presented an experimental evaluation on different versions of Snomed ct. The experiments suggest that, in both cases, a rigorous logic-based approach is computationally no more expensive than most ad-hoc approaches while providing significant advantages. In the case of logical diff, the advantage is that also subtle differences between TBoxes can be detected. In

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 3. Logical difference between semantic modules of two SNOMED CT versions

module extraction, our approach leads to smaller modules than many other approaches. It has to be noted, however, that so far our approach is limited to $\mathcal{EL}$. Extending at least some of the techniques to more expressive languages remains a challenging problem. For first results in this direction, we refer the reader to [7, 6].

References

1. F. Baader, C. Lutz, and B. Suntisrivaraporn. CEL—a polynomial-time reasoner for life science ontologies. In U. Furbach and N. Shankar, editors, *Proceedings of IJCAR'06*, volume 4130 of *LNAI*, pages 287–291. Springer-Verlag, 2006.
2. A. Borgida. On importing knowdledge from DL ontologies: some intuitions and problems. In *Proceedings of DL Workshop*, 2007.
3. P. Doran, V. Tamma, and L.Iannone. Ontology module extraction for ontology reuse:an ontology engineering perspective. In *Proceedings of CIKM*, 2007.
4. J. H. Gennari, M. A. Musen, R. W. Fergerson, W. E. Grosso, M. Crubézy, H. Eriksson, N. F. Noy, and S. W. Tu. The evolution of protégé: an environment for knowledge-based systems development. *Int. J. Hum.-Comput. Stud.*, 58(1):89–123, 2003.
5. B. C. Grau, I. Horrocks, Y. Kazakov, and U. Sattler. Just the right amount: extracting modules from ontologies. In *WWW*, pages 717–726, 2007.
6. B. Konev, C. Lutz, D. Walther, and F. Wolter. Semantic modularity and module extraction in description logic. manuscript, `http://www.csc.liv.ac.uk/~frank/publ/publ.html`.
7. B. Konev, D. Walther, and F. Wolter. The logical difference problem for description logic terminologies. manuscript, `http://www.csc.liv.ac.uk/~frank/publ/publ.html`.
8. C. Lutz, D. Walther, and F. Wolter. Conservative extensions in expressive description logics. In *Proceedings of IJCAI'07*. AAAI Press, 2007.
9. C. Lutz and F. Wolter. Conservative extensions in the lightweight description logic $\mathcal{EL}$. In *Proceedings of CADE'07*. Springer, 2007.
10. N. F. Noy and M. Musen. Promptdiff: A fixed-point algorithm for comparing ontology versions. In *Proceedings of AAAI*, pages 744–750, 2002.
11. J. Seidenberg and A. L. Rector. Web ontology segmentation: analysis, classification and use. In *WWW*, pages 13–22, 2006.
12. K. Spackman. Managing clinical terminology hierarchies using algorithmic calculation of subsumption: Experience with SNOMED-RT. *JAMIA*, 2000.
13. The Caml team. `http://caml.inria.fr/contact.en.html`.