Versioning Based on Logical Difference for Lightweight Description Logic Terminologies

William Gatens and Boris Konev and Michel Ludwig and Frank Wolter
Department of Computer Science, University of Liverpool, UK
w.gatens@student.liverpool.ac.uk, {konev, michel.ludwig, wolter}@liverpool.ac.uk

Abstract

We study a logic-based approach to versioning of ontologies. Under this view, ontologies provide answers to queries about some vocabulary of interest. The difference between two versions of an ontology is given by the set of queries that receive different answers. We investigate this approach for terminologies given in the description logic $\mathcal{EL}$ extended with role inclusions and range restrictions for three distinct types of queries: subsumption, instance, and conjunctive queries. We present an implementation, $\text{CEX}2$, that computes finite representations of the respective difference between terminologies.

1 Introduction

Terminologies are lightweight ontologies that are used to provide a common vocabulary for a domain of interest together with descriptions of the meaning of terms built from the vocabulary and relationships between them. They are being used in areas such as medical informatics, bio-informatics, and the semantic web to capture domain semantics and promote interoperability. Terminologies are often large and complex. For example, the widely used medical terminology SNOMED CT (Systematized Nomenclature of Medicine Clinical Terms) contains more than 300,000 term definitions ([IHTSDO], 2008). Another example is the National Cancer Institute ontology (NCI) consisting of more than 60,000 axioms [Golbeck et al., 2003]. Engineering, maintaining, and using such terminologies is a complex and laborious task, which is practically unfeasible without appropriate tool support. In this contribution, we focus on a principled logic-based approach to support for terminology versioning.

Dealing with multiple versions of the same information unit is nothing new in computing, and version control is a well established computer technology. Although modern version control systems provide a range of operations including support for collaborative development, branching, merging, etc., these operations extend and rely on the basic operations of detecting and representing the differences between versions. In this paper, we focus on this basic problem of versioning.

The need for versioning support is recognised by the ontology research community and ontology users, and a large number of approaches and tools have been developed. In our review of currently existing support for ontology versioning, we distinguish three approaches and describe them according to the difference between ontologies they compute:

1. versioning based on syntactic difference (syntactic diff);
2. versioning based on structural difference (structural diff);
3. versioning based on logical difference (logical diff).

The syntactic diff underlies most existing version control systems used in software development [Conradi and Westfechtel, 1998] (such as, for example, RCS, CVS, SCCS). It works with text files and represents the difference between versions as blocks of text present in one version but not another, ignoring any meta-information about the document. As observed already in [Noy and Musen, 2002], ontology versioning cannot rely on a purely syntactic diff operation since many syntactic differences (e.g., the order of ontology axioms) do not affect the semantics of ontologies. Therefore, ontology versioning based on syntactic difference is essentially limited to comparing rather informal change logs [Oliver et al., 1999].

The structural diff extends the syntactic diff by taking into account some structural meta-information about the distinct versions of the ontologies compared. It has been suggested for dealing with structured and hierarchical documents such as UML diagrams, database schemas, or XML documents (see e.g. [Ohst et al., 2003] and references within). For ontologies, the main characteristic of the structural diff is that it regards them as structured objects, such as an is-a taxonomy [Noy and Musen, 2002], a set of RDF triplets [Klein et al., 2002], or a set of class defining axioms [Redmond et al., 2008; Jimnez-Ruiz et al., 2011]. Changes to ontologies are mostly described in terms of structural operations, for example, adding or deleting a class, extending a class, renaming slots, moving a class from one place in the hierarchy to another, adding or deleting an axiom, class renaming, etc.; sometimes basic logical properties of ontologies, e.g., the equivalence of different structural forms of concepts, are also taken into account [Jimnez-Ruiz et al., 2011]. Ontology versioning based on structural diff of some form is available in most current ontology editors and ontology management systems either natively or through plugins [Noy and Musen, 2002; Klein et al., 2002; Jimnez-Ruiz et al., 2011].

Though very helpful, the structural diff still has the deficiency of having no unambiguous semantic foundation and
computing the instance- and conjunctive query-based logical and a plugin for Protégé. CEX2 also extends the functional-
tivity of the OwlDiff plugin [Kˇremen et al., 2008] and does not cover

The logical diff has only recently been introduced in
[Konev et al., 2008; Kontchakov et al., 2010] and completely abstracts from the representation of the ontology. Here, an ontology is regarded as a set of axioms formulated in a logical language with a formal and unambiguous semantics. Under this view, ontologies provide answers to queries about some vocabulary of interest. Typical queries include subsumption between concepts and, if the ontology is used to access instance data, instance checking and conjunctive query answering. The logical diff is motivated by this view. If two versions of an ontology give the same answers to a class of queries relevant to an application domain, they may be deemed to have no difference regardless of their syntactic or structural form; and queries producing different answers from the versions may be considered as a characterisation of the difference itself. In this way one can, for example, define exactly the differences visible when querying instance data or exactly the differences expressed by subsumptions between concepts.

To make this approach work in practice, at least two problems have to be solved:

- For many ontology languages and classes of queries the computational complexity of even detecting if two ontology versions differ is at least one exponential harder than ontology classification and is sometimes undecidable [Lutz et al., 2007; Lutz and Wolter, 2010; Grau et al., 2008].
- If the set of queries producing different answers from the two versions is not empty, it is typically infinite and, therefore, cannot be presented to the user as such. Thus, techniques to succinctly characterise its elements and present them to the user are required.

The aim of this paper is to provide solutions to these problems for terminologies given in the description logic $\mathcal{ELH}^t$ that extends the description $\mathcal{EL}$ underlying the OWL2 EL profile with role inclusions and domain and range restrictions [Baader et al., 2008]. This paper extends the results of [Konev et al., 2008] for “pure” $\mathcal{EL}$ by firstly handling those language extensions, vital for versioning SNOMED CT and NCI terminologies, and secondly by developing algorithms computing the instance- and conjunctive query-based logical diff. In contrast to $\mathcal{EL}$ itself, for the latter there are complex differences that are not visible when only subsumption queries are considered. We also present an extension, CEX2, of the CEX system introduced in [Konev et al., 2008] for computing the logical difference between $\mathcal{EL}$ terminologies and a plugin for Protégé. CEX2 also extends the functionality of the OwlDiff plugin [Kremen et al., ], which implements the algorithms from [Konev et al., 2008] and does not cover $\mathcal{ELH}^t$ nor instance differences.

### Table 1: Standard Translation

<table>
<thead>
<tr>
<th>Concept</th>
<th>Translation</th>
<th>Inclusion</th>
<th>Translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\top$</td>
<td>$x = x$</td>
<td>$\subseteq D$</td>
<td>$\forall x (C^D(x) \rightarrow D^D(x))$</td>
</tr>
<tr>
<td>$A$</td>
<td>$A(x)$</td>
<td>$\subseteq D$</td>
<td>$\forall x (C^D(x) \leftrightarrow D^D(x))$</td>
</tr>
<tr>
<td>$A \sqcap D$</td>
<td>$C^D(x) \land D^D(x)$</td>
<td>$r \subseteq s$</td>
<td>$\forall y (r(x, y) \rightarrow s(x, y))$</td>
</tr>
<tr>
<td>$\exists r.C$</td>
<td>$\exists y (r(x, y) \land C^D(y))$</td>
<td>$\exists$</td>
<td>$\exists$</td>
</tr>
<tr>
<td>$\text{dom}(r)$</td>
<td>$\exists y (r(x, y))$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>$\text{ran}(r)$</td>
<td>$\exists y (r(y, x))$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
</tr>
</tbody>
</table>

### 2 Preliminaries

Let $\mathcal{N}_C$, $\mathcal{N}_R$, and $\mathcal{N}_I$ be countably infinite and mutually disjoint sets of concept names, role names, and individual names. $\mathcal{EL}$-concepts $C$ are built according to the rule

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

where $A \in \mathcal{N}_C$, $r \in \mathcal{N}_R$, and $C, D$ range over $\mathcal{EL}$-concepts. The set of $\mathcal{ELH}^t$-inclusions consists of concept inclusions $C \subseteq D$ and concept equations $C \equiv D$, domain restrictions $\text{dom}(r) \subseteq C$, range restrictions $\text{ran}(r) \subseteq C$ and role inclusions $C \subseteq s$, where $C, D$ are $\mathcal{EL}$-concepts and $r, s \in \mathcal{N}_R$. An $\mathcal{ELH}^t$-TBox is a finite set of $\mathcal{ELH}^t$-inclusions. An $\mathcal{ELH}^t$-TBox is called an $\mathcal{ELH}^t$-terminology if all its concept inclusions and equations are of the form $A \subseteq C$ and $A \equiv C$ and no concept name occurs more than once on the left hand side.

In description logic, instance data are represented by $\text{ABox}$ assertions of the form $A(a)$ and $r(a, b)$, where $a, b \in \mathcal{N}_I$, $A \in \mathcal{N}_C$, and $r \in \mathcal{N}_R$. An $\text{ABox}$ is a finite set of $\text{ABox}$-assertions. By $\text{obj}(A)$ we denote the set of individual names in $A$. A knowledge base (KB) is a pair $(\mathcal{T}, A)$ consisting of a $\text{TBox}$ and an $\text{ABox}$. Assertions of the form $C(a)$ and $r(a, b)$, where $a, b \in \mathcal{N}_I$, $C$ an $\mathcal{EL}$-concept, and $r \in \mathcal{N}_R$, are called instance assertions. To define the semantics of $\mathcal{ELH}^t$ we make use of the fact that $\mathcal{ELH}^t$ expressions can be regarded as formulas in $\text{FO}$, where $\text{FO}$ denotes the set of first-order predicate logic formulas with equality using unary predicates in $\mathcal{N}_C$, binary predicates in $\mathcal{N}_R$, and constants from $\mathcal{N}_I$; see Figure 1. In what follows, we will not distinguish between $\mathcal{ELH}^t$ expressions and their translation into $\text{FO}$ and refer TBoxes, $\text{ABoxes}$ and KBs as finite subsets of $\text{FO}$. Thus, we use $\mathcal{T} \models \varphi$ to denote that $\varphi$ follows from $\mathcal{T}$ in first-order logic even if $\varphi$ is an $\mathcal{ELH}^t$-inclusion and $\mathcal{T}$ a subset of $\text{FO}$. $\text{FO}$ (and, therefore, $\mathcal{ELH}^t$) is interpreted in models $\mathcal{I} = (\Delta^2, \chi)$, where the domain $\Delta^2$ is a non-empty set, and $\chi$ is a function mapping each concept name $A$ to a subset $A^2$ of $\Delta^2$, each role name $r$ to a binary relation $r^2 \subseteq \Delta^2 \times \Delta^2$, and each individual name $a$ to an element $a^2 \in \Delta^2$.

The most important ways of querying $\mathcal{ELH}^t$-TBoxes and KBs are

- subsumption: check whether $\mathcal{T} \models \alpha$, for an $\mathcal{ELH}^t$-inclusion $\alpha$ and $\text{TBox}$ $\mathcal{T}$,
- instance checking: check whether $(\mathcal{T}, A) \models \alpha$ for an instance assertion $\alpha$ and $\text{KB}$ $(\mathcal{T}, A)$, and
- conjunctive query answering.

To define latter, call a first-order formula $q(\vec{z})$ a conjunctive query if it is of the form $\exists \vec{y} \psi(\vec{z}, \vec{y})$, where $\psi$ is a conjunction of expressions $A(t)$ and $r(t_1, t_2)$ with $t, t_1, t_2$ drawn from $\mathcal{N}_I$ and the sequences of variables $\vec{z}$ and $\vec{y}$. If $\vec{z}$ has length $k$, being syntax dependent. Moreover, it is tailored towards applications of ontologies which are based on the induced concept hierarchy (or some mild extension of it), but does not capture modern applications such as ontology-based data access (OBDA) [Poggi et al., 2008; Lutz et al., 2009] in which ontologies are used to provide a user-oriented view of the data and make it accessible via queries formulated solely in the language of the ontology without any knowledge of the actual structure of the data.
then a sequence $\bar{a}$ of elements of $\text{obj}(A)$ of length $k$ is called a certain answer to $q(x)$ of a KB $(\mathcal{T}, A)$ if $(\mathcal{T}, A) \models q(\bar{a})$.

### 3 Logical Difference

In this section, we introduce three notions of logical difference between TBoxes and the derived notion of $\Sigma$-inseparability. Intuitively, the logical difference between two TBoxes $\mathcal{T}_1$ and $\mathcal{T}_2$ should be the set of all ‘relevant formulas’ $\varphi$ such that $\mathcal{T}_1 \models \varphi$ and $\mathcal{T}_2 \not\models \varphi$ or vice versa. Of course, which formulas $\varphi$ are relevant depends on the application domain. In many applications only subsumptions between concepts are relevant, but if TBoxes are employed to access instance data, then answers to instance or even conjunctive queries can be relevant as well. In addition, in applications of large-scale terminologies such as SNOMED CT and NCI only a very small subset of the vocabulary of the terminologies is relevant. Thus, a meaningful notion of logical difference only a very small subset of the vocabulary of the terminology queries can be relevant as well. In addition, in applications of large-scale terminologies such as SNOMED CT and NCI only a very small subset of the vocabulary of the terminologies is relevant. Thus, a meaningful notion of logical difference.

#### Definition 3

Let $\mathcal{T}_1 = (A \sqsubseteq \exists r.B), \mathcal{T}_2 = 0, \Sigma = \{A, B\}$. Then $\mathcal{T}_1$ and $\mathcal{T}_2$ are $\Sigma$-instance inseparable, but they are not $\Sigma$-query inseparable. Consider the $\Sigma$-ABox $A = \{A(\bar{a})\}$ and the $\Sigma$-query $q = \exists x B(x)$. Then $(\mathcal{T}_1, A) \models q$ but $(\mathcal{T}_2, A) \not\models q$.

It is shown in [Lutz and Wolter, 2010] that Example 6 is essentially the only situation in which there is a difference between $\Sigma$-instance inseparability and $\Sigma$-query inseparability in $\mathcal{EL}$. The two notions become equivalent for $\mathcal{EL}$ if the universal role is admitted in instance queries (e.g., the conjunctive query $\exists x B(x)$ corresponds to the instance query $\exists a B(a)$ for the universal role $\rho$). In contrast, for $\mathcal{EL}^+$ there are more subtle differences between the instance and the query case.

#### Example 7

Let $\mathcal{T}_1 = (A \sqsubseteq \exists r.T, s \sqsubseteq r_1, s \sqsubseteq r_2), \mathcal{T}_2 = \{A \sqsubseteq \exists r_1. T \sqcap \exists r_2. T\}, \Sigma = \{A, r_1, r_2\}$. Then $\mathcal{T}_1$ and $\mathcal{T}_2$ are $\Sigma$-concept and $\Sigma$-instance inseparable, but they are not $\Sigma$-query inseparable. To show the latter, let $A = \{A(a)\}$ and let $q = \exists x (r_1(a, x) \land r_2(a, x))$. Then $(\mathcal{T}_1, A) \models q$ but $(\mathcal{T}_2, A) \not\models q$.

Note that one can easily show for all $\mathcal{EL}^+$ terminologies $\mathcal{T}_1$ and $\mathcal{T}_2$ and all signatures $\Sigma$:

$$\mathcal{T}_1 \equiv_T \mathcal{T}_2 \Rightarrow \mathcal{T}_1 \equiv_\Sigma \mathcal{T}_2 \Rightarrow \mathcal{T}_1 \equiv_\Sigma \mathcal{T}_2.$$  

Having introduced the three notions of logical difference we are interested in, we now face two problems: how to detect whether there is a difference between $\mathcal{EL}^+$ terminologies and, if so, how to represent the differences.

### 4 $\Sigma$-Concept Difference

We start with the concept difference. The following lemma states that for any $\varphi \in c\text{Diff}^2(\mathcal{T}_1, \mathcal{T}_2)$ one can find a ‘primitive’ representative $\varphi' \in c\text{Diff}^2(\mathcal{T}_1, \mathcal{T}_2)$ of $\varphi$.

#### Lemma 8

Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be $\mathcal{EL}^+$-terminologies and $\Sigma$ a signature. If $\varphi \in c\text{Diff}^2(\mathcal{T}_1, \mathcal{T}_2)$, then there exist either $\{r, s\} \subseteq \text{sig}(\varphi)$ with $r \sqsubseteq s \in c\text{Diff}^2(\mathcal{T}_1, \mathcal{T}_2)$ or one of

(i) $C \subseteq A$ or $A \subseteq D$

(ii) $\exists r.C \subseteq \exists r.A$

(iii) $\text{dom}(r) \subseteq D$ or $\text{ran}(r) \subseteq D$

is a member of $c\text{Diff}^2(\mathcal{T}_1, \mathcal{T}_2)$, where $r \in \text{sig}(\varphi), A \in \text{sig}(\varphi)$ is a concept name and $C, D$ are $\mathcal{EL}$ concepts occurring in $\varphi$: the size of $C$ and $D$ is at most exponential in $\mathcal{T}_1$ and $\mathcal{T}_2$.  

38

ARCOE-11 Workshop Notes
The bounds on the size of $C$ and $D$ are optimal: in Lemma 8, in the worst case one can only find concepts $C$ and $D$ of exponential size in $T_1$ and $T_2$. To obtain a finite, uniquely determined, and high-level representation of the $\Sigma$-concept difference, we call $x \in \Sigma$ a $\Sigma$-concept difference witness if either $x$ is a concept name $A$ satisfying condition (i) or condition (ii); or $x$ is a role name $r$ satisfying condition (iii). ($\Sigma$-differences of the form $r \subseteq s$ can be easily computed and represented, and will not be further analysed in this paper.)

**Theorem 9.** The set of all $\Sigma$-concept difference witnesses can be computed in polynomial time.

$\Sigma$-concept difference witnesses can be regarded as representatives of primitive $cDiff$ elements. In particular, if there are no difference witnesses and no role inclusions in $cDiff(T_1, T_2)$, then $cDiff(T_1, T_2) = \emptyset$ and we obtain:

**Corollary 10.** For $\mathcal{ELH}'$, $\Sigma$-inseparability is decidable in polynomial time.

Our proofs of these results extends the Gentzen-style sequent calculus of [Hofmann, 2005] for $\mathcal{EL}$ to $\mathcal{ELH}'$ and then employs the extension to analyse derivations in $\mathcal{ELH}'$.

## 5 Data-Observable Differences

As we have seen above, there are differences between TBoxes that are visible using ABoxes but not visible using $\mathcal{ELH}'$-inclusions. The following result shows how one can represent the additional members of the $\Sigma$-instance difference.

**Lemma 11.** If $(A, \alpha) \in dDiff_{\Sigma}(T_1, T_2)$, then

- $\varphi \in \mathop{dDiff}_{\Sigma}(T_1, T_2)$, for some $\mathcal{ELH}'$-inclusion $\varphi$ such that $\mathop{sig}(\varphi) \subseteq \mathop{sig}(\alpha)$ or $\mathop{sig}(\varphi) \ni (\mathop{obj}(\alpha))$
- $(A', \alpha(b)) \in \mathop{dDiff}_{\Sigma}(T_1, T_2)$, for some concept name $A \in \mathop{sig}(\alpha)$, $A' \subseteq A$ of polynomial size in $T_1$ and $T_2$ and some $b \in \mathop{obj}(\alpha)$.

Similarly to the $\Sigma$-concept difference case, call a concept name $A \in \Sigma$ satisfying Point 2 of Lemma 11 for some ABox $A'$ a $\Sigma$-instance difference witness.

**Theorem 12.** The set of all $\Sigma$-instance difference witnesses can be computed in polynomial time. In particular, for $\mathcal{ELH}'$, $\Sigma$-instance inseparability is decidable in polynomial time.

Note that, in addition, for any $\Sigma$-instance difference witness $A$, one can compute in polynomial time some $A'$ with $(A', A(b)) \in dDiff(T_1, T_2)$, for some $b$.

The following lemma shows how one can represent members of the $\Sigma$-query difference that are not members of the $\Sigma$-instance difference.

**Lemma 13.** If $(A, q(a)) \in qDiff_{\Sigma}(T_1, T_2)$, then

- $(A, \alpha) \in dDiff_{\Sigma}(T_1, T_2)$, for some instance assertion $\alpha$ with $\mathop{sig}(\alpha) \subseteq \mathop{sig}(q)$
- $(A', q'(b)) \in qDiff_{\Sigma}(T_1, T_2)$, for some ABox $A'$ of cardinality at most two and $\mathop{sig}(A') \subseteq \mathop{sig}(A)$, conjunctive query $q'(x)$ with $\mathop{sig}(q') \subseteq \mathop{sig}(q)$ and $b \in \mathop{obj}(A')$.

Call an ABox $A'$ satisfying Point 2 of Lemma 13 for some conjunctive query $q'(b)$ a $\Sigma$-query difference witness.

**Theorem 14.** The set of all $\Sigma$-query difference witnesses can be computed in polynomial time. In particular, for $\mathcal{ELH}'$, $\Sigma$-query inseparability is decidable in polynomial time.

The proofs of Lemmas 11 and 13 employ model-theoretic methods and the sequent calculus developed for $\mathcal{ELH}'$.

We close this section with the observation that tractability of deciding $\Sigma$-inseparability is extremely non-robust under extensions of the language. For example, one can show that the addition of disjointness assertions $A \cap B \subseteq \perp$, $A, B$ concept names, to $\mathcal{EL}$-terminologies makes the three inseparability notions discussed in this section NP-hard. Inseparability becomes undecidable if general role inclusions $r_1 \circ \ldots \circ r_n \subseteq r$ are added to $\mathcal{EL}$-terminologies. The complexity of inseparability for simpler conditions on roles (such as transitivity) is unknown.

## 6 Systems and Experiments

We have implemented polynomial time algorithms computing difference witnesses for $cDiff_{\Sigma}(T_1, T_2)$ and $dDiff_{\Sigma}(T_1, T_2)$ for $\mathcal{ELH}'$-terminologies in the CEX2 tool\(^1\). CEX2 also computes representatives, in the sense of Lemmas 8 and 11, of $cDiff_{\Sigma}(T_1, T_2)$ and $dDiff_{\Sigma}(T_1, T_2)$, respectively, which are invaluable for understanding the difference witnesses. Note that CEX2 currently cannot analyse the $\Sigma$-query difference $qDiff_{\Sigma}(T_1, T_2)$ between $\mathcal{ELH}'$-terminologies. CEX2 extends CEX [Konev et al., 2008], which can only enumerate $\Sigma$-concept difference witnesses for $\mathcal{EL}$-terminologies. The performance of CEX2 is similar to that of CEX, for an evaluation of which we refer to [Konev et al., 2008]. We applied CEX2 to compare the January 2009 ($SM_{09a}$) and July 2009 ($SM_{09b}$) versions of SNOMED CT. $SM_{09a}$ contains 310013 concept names and 62 role names, whereas $SM_{09b}$ contains 307693 concept names and the same role names. For our experiments we used 159 signatures ranging over so called SNOMED CT subsets used in the UK for the deployment of SNOMED CT in specific areas.

We illustrate the results on a typical subset called “Specimen Material Type”\(^2\). Its signature contains 6510 concept names shared with $SM_{09b}$, which we extended with all 62 SNOMED CT role names. The output of CEX2 shows that

(i) $cDiff_{\Sigma}(SM_{09b}, SM_{09a}) = dDiff_{\Sigma}(SM_{09b}, SM_{09a})$
(ii) there are no differences regarding role inclusions;
(iii) there are 10 concept names $A$ such that there are concept inclusions of the form $C \subseteq A$ contained in the set $cDiff_{\Sigma}(SM_{09b}, SM_{09a})$;
(iv) there are 46 concept names $A$ such that there are concept inclusions of the form $A \subseteq D$ in $cDiff_{\Sigma}(SM_{09b}, SM_{09a})$.

In Point (iii) and (iv), the longest representatives $C, D$ computed by CEX2 had twelve concept and role name occurrences (thus they were smaller than the exponential worst case suggests). The relatively small number of $\Sigma$-concept difference witnesses allows one to analyse the inclusions from

---

\(^1\)Available under an open-source license at http://www.csc.liv.ac.uk/~michel/arcoe11/

\(^2\)Additional experimental results can be found at http://www.csc.liv.ac.uk/~michel/arcoe11/
cDiff\(_\Sigma\)(SM\(_{09b}\), SM\(_{09a}\)) computed by CEX2 further. For example, to explain and understand why

\[
\text{VenipunctureForBloodTest} \sqsubseteq (\exists \text{roleGroup} . \exists \text{hasFocus} . \text{EvaluationProcedure})
\]

is a member of the \(\Sigma\)-difference, we have computed, using axiom pinpointing [Baader et al., 2007], a minimal set of axioms from SM\(_{09b}\) which entails the concept inclusion above; the set is shown in Fig. 2. Axioms 2 and 3 are in both terminologies, but SM\(_{09a}\) contains

\[
\text{LaboratoryTest} \sqsubseteq \text{LaboratoryProcedure}
\]

instead of Axiom 1. Concept and role names from \(\Sigma\) are shaded in gray. It can be seen that the interaction between \(\Sigma\)-concepts heavily depends on inclusions that are built up mainly from non-\(\Sigma\)-concepts; actually none of inclusions required to derive (*) is a \(\Sigma\)-inclusion.

CEX2 is a text-based tool. In order to make it more accessible to ontology users, a Protégé plugin, LogDiffViz\(^3\), was created, which calls CEX2 and visualises both ontology versions and the differences as a hierarchical structure. LogDiffViz also provides basic axiom pinpointing. The plugin is distributed as a self-contained Java archive file (JAR) in which CEX2 is bundled.

7 Conclusion

We have presented a theoretical foundation for ontology versioning based on the logical diff for \(\mathcal{EL}^H\) terminologies. The algorithms have been implemented in the CEX2 tool and applied to SNOMED CT. We facilitate the use of the CEX2 tool through a Protégé plugin. It would be of interest to extend the theory and implementation further so as to cover more expressive role inclusions (e.g., transitive roles).

References


\(^3\)Available from http://www.csc.liv.ac.uk/~cs8wg/LogDiffViz/


