A Practical Approach for Computing Generalization Inferences in $\mathcal{EL}$

Rafael Peñaloza and Anni-Yasmin Turhan

TU Dresden, Germany,
Institute of Theoretical Computer Science
email: *last name*@tcs.inf.tu-dresden.de

Abstract. We present methods that compute generalizations of concepts or individuals described in ontologies written in the Description Logic $\mathcal{EL}$. These generalizations are the basis of methods for ontology design and are the core of concept similarity measures. The reasoning service least common subsumer (lcs) generalizes a set of concepts. Similarly, the most specific concept (msc) generalizes an individual into a concept description. For $\mathcal{EL}$ with general $\mathcal{EL}$-TBoxes, the lcs and the msc may not exist. However, it is possible to find a concept description that is the lcs (msc) up to a certain role-depth.
In this paper we present a practical approach for computing the lcs and msc with a bounded depth, based on the polynomial-time completion algorithm for $\mathcal{EL}$ and describe its implementation.

1 Introduction

Ontologies have become a commonly used means to describe controlled vocabularies, most prominently, in life sciences. Categories that form these vocabularies are sometimes only described in terms of specializations, i.e. by the "is-a" relation. Since the standardization of the web ontology language OWL [25], more applications have begun using this richer modeling language for describing notions from their domain in a more precise and detailed way. The formalism underlying OWL are Description Logics (DLs) [3], which are a family of logics with formal semantics. The formal semantics of DLs are the basis for the definition of reasoning services such as *subsumption* or *instance checking*. Subsumption tests whether a sub- / super-concept relationship holds between a pair of concept descriptions. Instance checking answers the question whether it follows from the ontology that a given individual must belong to a concept. The reasoning algorithms for these reasoning services are well-investigated for a range of DLs and implemented in powerful reasoner systems. In this paper we want to devise computation methods for inferences that can be employed to derive generalizations. These inferences turn out to be useful for range of ontology-based applications such as e.g. the life sciences [21, 9] or context-aware systems [22].

The newest version of the OWL standard [25] offers several *OWL profiles*, which correspond to DLs with varying expressivity. We are interested in the OWL EL profile, which corresponds to the DL $\mathcal{EL}$++, an extension of the DL $\mathcal{EL}$ where reasoning is still tractable. $\mathcal{EL}$-concept descriptions are composed from conjunctions or existential restrictions. Despite its limited expressivity, $\mathcal{EL}$ has turned out to be useful to model notions from life science applications. Most prominently, the medical ontology SnoMed [21] and the Gene Ontology [9] are written in $\mathcal{EL}$. For instance, it is possible to express by

$$\mathsf{Myocarditis} \sqsubseteq \mathsf{inflammation} \sqcap \exists \mathsf{has-location.heart}$$

that myocarditis is a kind of inflammation that is located in the heart.

In fact, medical and context-aware applications deal with very large ontologies, which are often *light-weight*, in the sense that they can be formulated in $\mathcal{EL}$ or one of its extensions from the so-called $\mathcal{EL}$-family. Members of the $\mathcal{EL}$-family allow for reasoning in polynomial time [2]. In particular, subsumption and instance checking are tractable for $\mathcal{EL}$ and $\mathcal{EL}$++, which was the main reason to standardize it in an own OWL 2 profile [25]. The reasoning algorithms for the $\mathcal{EL}$-family are based on a completion method and have been implemented in optimized reasoners such as CEL [16].

We investigate here two inferences that generalize different entities from DL knowledge bases. The first one is the *least common subsumer* (lcs) [7], which generalizes a collection of concept descriptions into a single concept description that is the least w.r.t. subsumption. Intuitively, the lcs yields a new (complex) concept description that captures all the commonalities of the input concept descriptions. The second inference is the *most specific concept* (msc) [4], which generalizes an individual into a concept description. Intuitively, the msc delivers the most specific concept description that is capable of describing the individual.

Applying Generalization Inferences

In the following we describe some of the most prominent applications of the lcs and the msc.

Similarity measures. Concept similarity measures compute, given a pair of concept descriptions, a numerical value between 0 and 1 that lies closer to 1 the more similar the concepts are. Similarity measures are an important means to discover, for instance, functional similarities of genes modeled in ontologies. In [13] and, more recently, in [19] several similarity measures were evaluated for the Gene Ontology and it was concluded that the similarity measure from Resnik [20] performed well, if not best. This similarity measure is an edge-based approach, which finds the most specific common ancestor (msa) [1] of the concepts to be compared in the concept hierarchy and computes a similarity value based on the number of edges between the concepts in question and their msa. Clearly, the msa can only yield a named concept from the TBox and thus captures possibly only *some* of the commonalities of the concepts to be compared. The lcs, in contrast, captures *all* commonalities and is thus a more faithful starting point for a similarity measure. In fact, the lcs was employed for similarity measures for DLs in [6] already. In a similar fashion a similarity measure for comparing individuals can be based on the msc [10]

[1] Sometimes also called *least common ancestor* (lca)

Building ontologies. In [11] it was observed that users working with biological ontologies would like to develop the description of the application categories in an example-driven way. More precisely, users would like to start by modeling individuals which are then generalized into a concept description. In fact, in the bottom-up approach for the construction of knowledge bases [4], a collection of individuals is selected for which a new concept definition is to be introduced in the ontology. Such a definition can be generated automatically by first generalizing each selected individual into a concept description (by computing the msc for each of them) and then applying the lcs to these concept descriptions.

The lcs can also be employed to enrich unbalanced concept hierarchies by adding new intermediate concepts [23].

Reconciling heterogeneous sources. The bottom-up procedure sketched before can also be employed in applications that face the problem that different information sources provide differing observations for the same state of affairs. For instance, in context-aware systems a GPS sensor or a video camera can provide differing information on a the location of a user. Alternatively, in medical applications, different diagnosing methods may yield differing results. It can be determined what the different sources agree on by representing this information as distinct ABox individuals and then by finding a common generalization of them by the bottom-up approach.

Information retrieval. The msc inference can be employed to obtain a query concept from an individual to search for other, similar individuals in an ontology [15, 8].

In order to support all these ontology services for practical applications automatically, computation algorithms for the generalization inferences in $\mathcal{EL}$ are needed. Unfortunately, the lcs in $\mathcal{EL}$ does not always exist, when computed w.r.t. cyclic TBoxes [1]. Similarly, the msc in $\mathcal{EL}$ does not always exist, if the ABox is cyclic [12], mainly because cyclic structures cannot be captured in $\mathcal{EL}$-concept descriptions. In [12] the authors propose to use an approximation of the msc by limiting the role-depth of the concept description computed. We pursue this approach here for the lcs and the msc and thus would obtain only "common subsumers" and "specific concepts" that are still generalizations of the input, but not necessarily the least ones w.r.t. subsumption. However, by our proposed method we obtain *the* lcs or *the* msc w.r.t. the given role depth bound. We argue that such approximations are still useful in practice.

Recently, a different approach for obtaining the lcs (or the msc) in presence of cyclic knowledge bases was proposed in [14] by extending $\mathcal{EL}$ with concept constructors for greatest fixpoints. In the so obtained DL $\mathcal{EL}^{\nu}$ reasoning stays polynomial and the lcs and msc w.r.t. cyclic knowledge bases can be computed. However, the DL obtained by adding constructors for greatest fixpoints is possibly not easy to comprehend for naive users of ontologies.

For medical or context-aware applications knowledge bases can typically grow very large in practice. Thus, in order to support the computation of the (role-depth bounded) lsc or the msc for such applications, efficient computation of these generalizations for $\mathcal{EL}$ is desirable. Our computation methods build directly on the completion method for subsumption and instance checking for $\mathcal{EL}$ [2] for which optimizations already exists and are employed in modern reasoner systems. This enables the implementation of the role-depth bounded lcs and msc on top of existing reasoner systems. More precisely, in our completion-based approach, we obtain the role-depth bounded lcs by traversing the data-structures built during the computation of the subsumption hierarchy of the ontology. The role-depth bounded msc can be obtained from the data-structures generated during the computation of all instance relations for the knowledge base. We have recently implemented the completion-based computation of the role-depth bounded lcs and msc in our system GEL.

This paper is structured as follows: after introducing basic notions of DLs, we discuss the completion algorithms for classification and instance checking in $\mathcal{EL}$ in Section 3. We extend these methods to computation algorithms for the role-depth bounded lcs in Section 4.1 and for the role-depth bounded msc in Section 4.2 and we describe our initial implementation of the presented methods in Section 5. We conclude the paper with an outline of possible future work.

2 Preliminaries

We now formally introduce the DL $\mathcal{EL}$. Let N_I, N_C and N_R be disjoint sets of *individual names*, *concept names* and *role names*, respectively. *$\mathcal{EL}$-concept descriptions* are built according to the syntax rule

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

where $A \in N_C$, and $r \in N_R$.

A *general concept inclusion* (GCI) is a statement of the form $C \sqsubseteq D$, where C, D are $\mathcal{EL}$- concept descriptions. An *$\mathcal{EL}$-TBox* is a finite set of GCIs. Observe that TBoxes can be cyclic and allow for multiple inheritance. An *$\mathcal{EL}$-ABox* is a set of assertions of the form $C(a)$, or $r(a, b)$, where C is an $\mathcal{EL}$-concept description, $r \in N_R$, and $a, b \in N_I$. An *ontology* or *knowledge base* $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ consists of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$.

The semantics of $\mathcal{EL}$ is defined by means of interpretations $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ consisting of a non-empty *domain* $\Delta^\mathcal{I}$ and an *interpretation function* $\cdot^\mathcal{I}$ that assigns binary relations on $\Delta^\mathcal{I}$ to role names, subsets of $\Delta^\mathcal{I}$ to concepts and elements of $\Delta^\mathcal{I}$ to individual names. The interpretation function $\cdot^\mathcal{I}$ is extended to concept descriptions in the usual way. For a more detailed description of the semantic of DLs see [3].

An interpretation $\mathcal{I}$ *satisfies* a concept inclusion $C \sqsubseteq D$, denoted as $\mathcal{I} \models C \sqsubseteq D$ if $C^\mathcal{I} \subseteq D^\mathcal{I}$; it *satisfies* an assertion $C(a)$ (or $r(a, b)$), denoted as $\mathcal{I} \models C(a)$ ($\mathcal{I} \models r(a, b)$, resp.) if $a^\mathcal{I} \in C^\mathcal{I}$ ($(a^\mathcal{I}, b^\mathcal{I}) \in r^\mathcal{I}$, resp.). An interpretation $\mathcal{I}$ is a *model* of a knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ if it satisfies all GCIs in $\mathcal{T}$ and all assertions in $\mathcal{A}$.

We say that C is *subsumed* by D w.r.t. $\mathcal{T}$ (written $C \sqsubseteq_\mathcal{T} D$) if for every model $\mathcal{I}$ of $\mathcal{T}$ it holds that $\mathcal{I} \models C \sqsubseteq D$. The computation of the subsumption hierarchy of all named concepts in a TBox is called *classification*.

Finally, an individual $a \in N_I$ is an *instance* of a concept description C w.r.t. $\mathcal{K}$ (written $\mathcal{K} \models C(a)$) if $\mathcal{I} \models C(a)$ for all models $\mathcal{I}$ of $\mathcal{K}$. *ABox realization* is the task of computing, for each individual a in $\mathcal{A}$, the set of named concepts from $\mathcal{K}$ that have a as an instance and that are least (w.r.t. $\sqsubseteq$).

In this paper we are interested in computing generalizations by least common subsumers and most specific concepts, which we now formally define. Notice that our definition is general for any DL and not necessarily specific for $\mathcal{EL}$.

Definition 1 (least common subsumer). *Let $\mathcal{L}$ be a DL, $\mathcal{K}$ = ($\mathcal{T}$, $\mathcal{A}$) be a $\mathcal{L}$-KB. The* least common subsumer *(lcs) w.r.t. $\mathcal{T}$ of a collection of concepts $C_1, \dots, C_n$ is the $\mathcal{L}$-concept description C such that*

1. *$C_i \sqsubseteq_\mathcal{T} C$ for all $1 \leq i \leq n$, and*
2. *for each $\mathcal{L}$-concept description D holds: if $C_i \sqsubseteq_\mathcal{T} D$ for all $1 \leq i \leq n$, then $C \sqsubseteq_\mathcal{T} D$.*

We will mostly consider the DL $\mathcal{EL}$ in this paper. Although defined as an n-ary operation, we will often write the lcs as a binary operation in the remainder of the paper for simplicity.

Definition 2 (most specific concept). *Let $\mathcal{L}$ be a DL, $\mathcal{K}$ = ($\mathcal{T}$, $\mathcal{A}$) be a $\mathcal{L}$-KB. The* most specific concept *(msc) w.r.t. $\mathcal{K}$ of an individual a from $\mathcal{A}$ is the $\mathcal{L}$-concept description C such that*

1. *$\mathcal{K} \models C(a)$, and*
2. *for each $\mathcal{L}$-concept description D holds: $\mathcal{K} \models D(a)$ implies $C \sqsubseteq_\mathcal{T} D$.*

Both inferences depend on the DL in use. For the DLs with conjunction as concept constructor the lcs and msc are, if exist, unique up to equivalence. Thus it is justified to speak of *the* lcs or *the* msc. Our computation methods for generalizations are based on the completion method, which we introduce in the following section.

3 Completion Algorithms for $\mathcal{EL}$

In principle, completion algorithms try to construct minimal models of the knowledge base. In case of classification algorithms such a model is constructed for the TBox and in case of ABox realization for the whole knowledge base. We describe the completion algorithm for ABox realization in $\mathcal{EL}$, originally described in [2], which can be easily restricted to obtain algorithms for classification. While the former is the basis for computing the role-depth bounded msc, the latter is used to obtain the role-depth bounded lcs.

For an $\mathcal{EL}$-KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ we want to test whether $\mathcal{K} \models D(a)$ holds. The completion algorithm first adds to $\mathcal{K}$ a concept name for the complex concept description D used in the instance check, i.e., $\mathcal{K} = (\mathcal{T} \cup \{A_q \equiv D\}, \mathcal{A})$, where A_q is a fresh concept name in $\mathcal{K}$. The instance checking algorithm for $\mathcal{EL}$ normalizes the knowledge base in two steps: first the ABox is transformed into a simple ABox. An ABox is a *simple ABox*, if it only contains concept names in concept assertions. An $\mathcal{EL}$-ABox $\mathcal{A}$ can be transformed into a simple ABox by first replacing each complex assertion $C(A)$ in $\mathcal{A}$ by $A(a)$ with a fresh name A and, second, introduce $A \equiv C$ in the TBox.

To describe the second normalization step, we need some notation. Let X be a concept description, a TBox, an ABox or a knowledge base. $\mathsf{CN}(X)$ denotes the set

$$\begin{array}{ll}
\textbf{NF1} & C \sqcap \hat{D} \sqsubseteq E \longrightarrow \{\ \hat{D} \sqsubseteq A, C \sqcap A \sqsubseteq E\ \} \\
\textbf{NF2} & \exists r.\hat{C} \sqsubseteq D \longrightarrow \{\ \hat{C} \sqsubseteq A, \exists r.A \sqsubseteq D\ \} \\
\textbf{NF3} & \hat{C} \sqsubseteq \hat{D} \longrightarrow \{\ \hat{C} \sqsubseteq A, A \sqsubseteq \hat{D}\ \} \\
\textbf{NF4} & B \sqsubseteq \exists r.\hat{C} \longrightarrow \{\ B \sqsubseteq \exists r.A, A \sqsubseteq \hat{C}\ \} \\
\textbf{NF5} & B \sqsubseteq C \sqcap D \longrightarrow \{\ B \sqsubseteq C, B \sqsubseteq D\ \}
\end{array}$$

where $\hat{C}, \hat{D} \notin \mathsf{CN}(\mathcal{T}) \cup \{\top\}$ and A is a new concept name.

Fig. 1. $\mathcal{EL}$ normalization rules

of all concept names and $\mathsf{RN}(X)$ denotes the set of all role names that appear in X. The *signature of* X (denoted $\mathsf{sig}(X)$) is then $\mathsf{CN}(X) \cup \mathsf{RN}(X)$. Now, an $\mathcal{EL}$-TBox $\mathcal{T}$ is in *normal form* if all concept axioms have one of the following forms, where $C_1, C_2 \in \mathsf{sig}(\mathcal{T})$ and $D \in \mathsf{sig}(\mathcal{T}) \cup \{\bot\}$:

$$C_1 \sqsubseteq D, \quad C_1 \sqcap C_2 \sqsubseteq D, \quad C_1 \sqsubseteq \exists r.C_2 \quad \text{or} \quad \exists r.C_1 \sqsubseteq D.$$

Any $\mathcal{EL}$-TBox can be transformed into normal form by introducing new concept names and by simply applying the normalization rules displayed in Figure 1 exhaustively. These rules replace the GCI on the left-hand side of the rules with the set of GCIs on the right-hand side.

Clearly, for a KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ the signature of $\mathcal{A}$ may be changed only during the first of the two normalization steps and the signature of $\mathcal{T}$ may be extended during both of the normalization steps. The normalization of the KB can be done in linear time.

The completion algorithm for instance checking is based on the one for classifying $\mathcal{EL}$-TBoxes introduced in [2]. Let $\mathcal{K}$ =($\mathcal{T}$, $\mathcal{A}$) be a normalized $\mathcal{EL}$-KB, i.e., with a simple ABox $\mathcal{A}$ and a TBox $\mathcal{T}$ in normal form. The completion algorithm works on four kinds of *completion sets*: $S(a), S(a,r), S(C)$ and $S(C,r)$ for each $a \in \mathsf{IN}(\mathcal{A})$, each $C \in \mathsf{CN}(\mathcal{K})$, and each $r \in \mathsf{RN}(\mathcal{K})$. The sets of the kind $S(a)$ and $S(a,r)$ contain individuals and concept names. The completion algorithm for classification uses only the latter two kinds of completion sets: $S(C)$ and $S(C,r)$, which contain only concept names from $\mathsf{CN}(\mathcal{K})$. Intuitively, the completion rules make implicit subsumption and instance relationships explicit in the following sense:

– $D \in S(C)$ implies that $C \sqsubseteq_{\mathcal{T}} D$,
– $D \in S(C,r)$ implies that $C \sqsubseteq_{\mathcal{T}} \exists r.D$.
– $D \in S(a)$ implies that a is an instance of D w.r.t. $\mathcal{K}$,
– $D \in S(a,r)$ implies that a is an instance of $\exists r.D$ w.r.t. $\mathcal{K}$.

$\mathsf{S}_{\mathcal{K}}$ denotes the set of all completion sets of a normalized $\mathcal{K}$. The completion sets are initialized for each $C \in \mathsf{CN}(\mathcal{K})$, each $r \in \mathsf{RN}(\mathcal{K})$, and each $a \in \mathsf{IN}(\mathcal{A})$ as follows:

– $S(C) := \{C, \top\}$
– $S(C,r) := \emptyset$
– $S(a) := \{C \in \mathsf{CN}(\mathcal{A}) \mid C(a) \text{ appears in } \mathcal{A}\} \cup \{\top\}$
– $S(a,r) := \{b \in \mathsf{IN}(\mathcal{A}) \mid r(a,b) \text{ appears in } \mathcal{A}\}$.

CR1 If $C \in S(X)$, $C \sqsubseteq D \in \mathcal{T}$, and $D \notin S(X)$
then $S(X) := S(X) \cup \{D\}$

CR2 If $C_1, C_2 \in S(X)$, $C_1 \sqcap C_2 \sqsubseteq D \in \mathcal{T}$, and $D \notin S(X)$
then $S(X) := S(X) \cup \{D\}$

CR3 If $C \in S(X)$, $C \sqsubseteq \exists r.D \in \mathcal{T}$, and $D \notin S(X, r)$
then $S(X, r) := S(X, r) \cup \{D\}$

CR4 If $Y \in S(X, r)$, $C \in S(Y)$, $\exists r.C \sqsubseteq D \in \mathcal{T}$, and
$D \notin S(X)$ then $S(X) := S(X) \cup \{D\}$

Fig. 2. $\mathcal{EL}$ completion rules

Then these sets are extended by applying the completion rules shown in Figure 2 until no more rule applies. In these rules C, C_1, C_2 and D are concept names and r is a role name, while X and Y can refer to concept or individual names in the algorithm for instance checking. In the algorithm for classification, X and Y refer to concept names. After the completion has terminated, the following relations hold between an individual a, a role r and named concepts A and B:

– subsumption relation between A and B from $\mathcal{K}$ holds iff $B \in S(A)$
– instance relation between a and B from $\mathcal{K}$ holds iff $B \in S(a)$,

which has been shown in [2].

To decide the initial query: $\mathcal{K} \models D(a)$, one has to test now, whether A_q appears in $S(a)$. In fact, instance queries for all individuals and all named concepts from the KB can be answered now; the completion algorithm does not only perform one instance check, but complete ABox realization. The completion algorithm for $\mathcal{EL}$ runs in polynomial time in size of the knowledge base.

4 Computing Role-depth Bounded Generalizations

We employ the completion method now to compute first the role-depth bounded lcs and then the role-depth bounded msc in $\mathcal{EL}$.

4.1 Computing the Role-depth Bounded LCS

As mentioned in the introduction, the lcs does not need to exist for cyclic TBoxes. Consider the TBox $\mathcal{T} = \{A \sqsubseteq \exists r.A \sqcap C,\ B \sqsubseteq \exists r.B \sqcap C\}$. The lcs of A and B is then

$$C \sqcap \exists r.(C \sqcap \exists r.(C \sqcap \exists r.(C \sqcap \cdots$$

and cannot be expressed by a finite concept description. To avoid such infinite nestings, we limit the role-depth of the concept description to be computed. The *role-depth* of a concept description C (denoted $rd(C)$) is the maximal number of nested quantifiers of C. Now we can define the lcs with limited role-depth.

Definition 3 (Role-depth bounded $\mathcal{L}$-lcs). *Let $\mathcal{T}$ be an $\mathcal{L}$-TBox and $C_1, \ldots, C_n$ $\mathcal{L}$-concept descriptions and $k \in \mathbb{N}$. Then the $\mathcal{L}$-concept description C is the* role-depth bounded $\mathcal{L}$-least common subsumer *of $C_1, \ldots, C_n$ w.r.t. $\mathcal{T}$ and role-depth k (written k-$lcs(C_1, \ldots, C_n)$) iff*

1. *$rd(C) \leq k$,*
2. *$C_i \sqsubseteq_\mathcal{T} C$ for all $1 \leq i \leq n$, and*
3. *for each $\mathcal{L}$-concept descriptions D with $rd(D) \leq k$ it holds that,*
   $C_i \sqsubseteq_\mathcal{T} D$ for all $1 \leq i \leq n$ implies $C \sqsubseteq_\mathcal{T} D$.

The computation algorithm for the role-depth bounded lcs w.r.t. general $\mathcal{EL}$-TBoxes, constructs the concept description from the set of completion sets. More precisely, it combines and intersects the completion sets in the same fashion as in the cross-product computation in the lcs algorithm for $\mathcal{EL}$-concept descriptions (without TBoxes) from [4]. The method we present here to compute the role-depth bounded lcs was described in [17].

However, the completion sets may contain concept names that were introduced during normalization. The returned lcs-concept description should only contain concept names that appear in the initial TBox, thus we need to "de-normalize" the concept descriptions obtained from the completion sets. However, the extension of the signature by normalization according to the normalization rules from Figure 1 does not affect subsumption tests for $\mathcal{EL}$-concept descriptions formulated w.r.t. the initial signature of $\mathcal{T}$. The following Lemma has been shown in [17].

Lemma 1. *Let $\mathcal{T}$ be an $\mathcal{EL}$-TBox and $\mathcal{T}'$ the TBox obtained from $\mathcal{T}$ by applying the $\mathcal{EL}$ normalization rules, C, D be $\mathcal{EL}$-concept descriptions with $\mathsf{sig}(C) \subseteq \mathsf{sig}(\mathcal{T})$ and $\mathsf{sig}(D) \subseteq \mathsf{sig}(\mathcal{T}')$ and D' be the concept description obtained by replacing all names $A \in \mathsf{sig}(\mathcal{T}') \setminus \mathsf{sig}(\mathcal{T})$ from D with $\top$. Then $C \sqsubseteq_{\mathcal{T}'} D$ iff $C \sqsubseteq_\mathcal{T} D'$.*

Lemma 1 guarantees that subsumption relations w.r.t. the normalized TBox $\mathcal{T}'$ between C and D, also hold w.r.t. the original TBox $\mathcal{T}$ for C and D', which is basically obtained from D by removing the names introduced by normalization, i.e., concept names from $\mathsf{sig}(\mathcal{T}') \setminus \mathsf{sig}(\mathcal{T})$.

We assume that the role-depth of each input concept of the lcs has a role-depth less or equal to k. This assumption is motivated by the applications of the lcs on the one hand and on the other by the simplicity of presentation, rather than a technical necessity. The algorithm for computing the role-depth bounded lcs of two $\mathcal{EL}$-concept descriptions is depicted in Algorithm 1.

The procedure k-lcs first adds concept definitions for the input concept descriptions to (a copy of) the TBox and transforms this TBox into the normalized TBox $\mathcal{T}'$. Next, it calls the procedure apply-completion-rules, which applies the $\mathcal{EL}$ completion rules exhaustively to the TBox $\mathcal{T}'$, and stores the obtained set of completion sets in S. Then it calls the function k-lcs-r with the concept names A and B for the input concepts, the set of completion sets S, and the role-depth limit k. The result is then de-normalized and returned (lines 4 to 6). More precisely, in case a complex concept description is returned from k-lcs-r, the procedure remove-normalization-names removes concept names that were added during the normalization of the TBox.

Algorithm 1 Computation of a role-depth bounded $\mathcal{EL}$-lcs.

```
Procedure k-lcs $(C, D, \mathcal{T}, k)$
Input: $C, D$: $\mathcal{EL}$-concept descriptions; $\mathcal{T}$: $\mathcal{EL}$-TBox; $k$: natural number
Output: $k\text{-}lcs(C, D)$: role-depth bounded $\mathcal{EL}$-lcs of $C$ and $D$ w.r.t $\mathcal{T}$ and $k$.
1: $\mathcal{T}'$ := normalize($\mathcal{T} \cup \{A \equiv C, B \equiv D\}$)
2: $\mathsf{S}_{\mathcal{T}'}$ := apply-completion-rules($\mathcal{T}'$)
3: $L$ := k-lcs-r $(A, B, \mathsf{S}_{\mathcal{T}'}, k)$
4: if $L = A$ then return $C$
5: else if $L = B$ then return $D$
6: else return remove-normalization-names($L$)
7: end if

Procedure k-lcs-r ($A$, $B$, S, $k$)
Input: $A, B$: concept names; S: set of completion sets; $k$: natural number
Output: $k\text{-}lcs(A, B)$: role-depth bounded $\mathcal{EL}$-lcs of $A$ and $B$ w.r.t $\mathcal{T}$ and $k$.
1: if $B \in S(A)$ then return $B$
2: else if $A \in S(B)$ then return $A$
3: end if
4: common-names := $S(A) \cap S(B)$
5: if $k = 0$ then return $\bigsqcap_{P \in \text{common-names}} P$
6: else return $\bigsqcap_{P \in \text{common-names}} P \sqcap$
       $\bigsqcap_{r \in \mathsf{RN}(\mathcal{T})} \Big( \bigsqcap_{(E,F) \in S(A,r) \times S(B,r)} \exists r.\ \text{k-lcs-r}\ (E, F, \mathsf{S}, k-1) \Big)$
7: end if
```

The function k-lcs-r gets a pair of concept names, a set of completion sets and a natural number as inputs. First, it tests whether one of the input concepts subsumes the other w.r.t. $\mathcal{T}'$. In that case the name of the subsuming concept is returned. Otherwise the set of concept names that appear in the completion sets of both input concepts is stored in common-names (line 4).[2] In case the role-depth bound is reached ($k = 0$), the conjunction of the elements in common-names is returned. Otherwise, the elements in common-names are conjoined with a conjunction over all roles $r \in \mathsf{RN}(\mathcal{T})$, where for each r and each element of the cross-product over the r-successors of the current A and B a recursive call to k-lcs-r is made with the role-depth bound reduced by 1 (line 6). This conjunction is then returned to k-lcs.

For $L = \text{k-lcs}(C, D, \mathcal{T}, k)$ it holds by construction that $rd(L) \leq k$.[3] We now show that the result of the function k-lcs is a common subsumer of the input concept descriptions. It was shown in [17] that all conditions of Definition 3 are fulfilled for k-lcs$(C, D, \mathcal{T}, k)$.

Theorem 1. *Let C and D be $\mathcal{EL}$-concept descriptions, $\mathcal{T}$ an $\mathcal{EL}$-TBox, $k \in \mathbb{N}$, then* k-lcs$(C, D, \mathcal{T}, k) \equiv k\text{-}lcs(C, D)$.

[2] Note, that the intersection $S(A) \cap S(B)$ is never empty, since both sets contain $\top$.
[3] Recall our assumption: the role-depth of each input concept is less or equal to k.

For cases where k-lcs returns a concept description with role-depth of less than k we conjecture that it is the exact lcs.

The complexity of the overall method is exponential. However, if a compact representation of the lcs with structure sharing is used, the lcs-concept descriptions can be represented polynomially.

If a k-lcs is too general and a bigger role depth of the k-lcs is desired, the completion of the TBox does not have to be redone for a second computation. The completion sets can simply be "traversed" further.

4.2 Computing the Role-depth Bounded MSC

The msc was first investigated for $\mathcal{EL}$-concept descriptions and w.r.t. unfoldable TBoxes and possibly cyclic ABoxes in [12]. Similar to the lcs, the msc does not need to exist, since cyclic structures cannot be expressed by $\mathcal{EL}$-concept descriptions. Now we can define the msc with limited role-depth.

Definition 4 (role-depth bounded $\mathcal{L}$-msc). *Let $\mathcal{K}$ =($\mathcal{T}$, $\mathcal{A}$) be a $\mathcal{L}$-KB and a an individual in $\mathcal{A}$ and $k \in \mathbb{N}$. Then the $\mathcal{L}$-concept description C is the* role-depth bounded $\mathcal{EL}$-most specific concept *of a w.r.t. $\mathcal{K}$ and role-depth k (written k-$msc_{\mathcal{K}}(a)$) iff*

1. *$rd(C) \leq k$,*
2. *$\mathcal{K} \models C(a)$, and*
3. *for each $\mathcal{EL}$-concept description D with $rd(D) \leq k$ holds: $\mathcal{K} \models D(a)$ implies $C \sqsubseteq_{\mathcal{T}} D$.*

In case the exact msc has a role-depth less than k the role-depth bounded msc is the exact msc.

Again, we construct the msc by traversing the completion sets to "collect" the msc. More precisely, the set of completion sets encodes a graph structure, where the sets $S(X)$ are the nodes and the sets $S(X, r)$ encode the edges. Traversing this graph structure, one can construct an $\mathcal{EL}$-concept. To obtain a finite concept in the presence of cyclic ABoxes or TBoxes one has to limit the role-depth of the concept to be obtained.

Definition 5 (traversal concept). *Let $\mathcal{K}$ be an $\mathcal{EL}$-KB, $\mathcal{K}''$ be its normalized form, $\mathsf{S}_{\mathcal{K}}$ the completion set obtained from $\mathcal{K}$ and $k \in \mathbb{N}$. Then the* traversal concept of a named concept *A (denoted k-$\mathsf{C}_{\mathsf{S}_{\mathcal{K}}}(A)$) with $\mathsf{sig}(A) \subseteq \mathsf{sig}(\mathcal{K}'')$ is the concept obtained from executing the procedure call* traversal-concept-c*(A, $\mathsf{S}_{\mathcal{K}}$, k) shown in Algorithm 2.*

The traversal concept of an individual *a (denoted k-$\mathsf{C}_{\mathsf{S}_{\mathcal{K}}}(a)$) with $a \subseteq \mathsf{sig}(\mathcal{K})$ is the concept description obtained from executing the procedure call* traversal-concept-i*(a, $\mathsf{S}_{\mathcal{K}}$, k) shown in Algorithm 2.*

The idea is that the traversal concept of an individual yields its msc. However, the traversal concept contains names that were introduced during normalization. The returned msc should be formulated w.r.t. the signature of the original KB, thus the normalization names need to be removed or replaced.

Algorithm 2 Computation of a role-depth bounded $\mathcal{EL}$-msc.

```
Procedure k-msc $(a, \mathcal{K}, k)$
Input: $a$: individual from $\mathcal{K}$; $\mathcal{K}$ =($\mathcal{T}$, $\mathcal{A}$) an $\mathcal{EL}$-KB; $k \in \mathbb{N}$
Output: role-depth bounded $\mathcal{EL}$-msc of $a$ w.r.t. $\mathcal{K}$ and $k$.
 1: $(\mathcal{T}', \mathcal{A}')$ := simplify-ABox($\mathcal{T}$, $\mathcal{A}$)
 2: $\mathcal{K}''$ := (normalize($\mathcal{T}'$), $\mathcal{A}'$)
 3: $\mathsf{S}_\mathcal{K}$ := apply-completion-rules($\mathcal{K}$)
 4: return Remove-normalization-names ( traversal-concept-i($a, \mathsf{S}_\mathcal{K}, k$))

Procedure traversal-concept-i ($a$, S, $k$)
Input: $a$: individual name from $\mathcal{K}$; S: set of completion sets; $k \in \mathbb{N}$
Output: role-depth traversal concept (w.r.t. $\mathcal{K}$) and $k$.
 1: if $k = 0$ then return $\bigsqcap_{A \,\in\, S(a)} A$
 2: else return $\bigsqcap_{A \,\in\, \mathsf{S}(a)} A \sqcap$
        $\bigsqcap_{r \in \mathsf{RN}(\mathcal{K}'')} \bigsqcap_{A \,\in\, \mathsf{CN}(\mathcal{K}'') \cap S(a,r)} \exists r.$ traversal-concept-c $(A, \mathsf{S}, k-1) \sqcap$
        $\bigsqcap_{r \in \mathsf{RN}(\mathcal{K}'')} \bigsqcap_{b \,\in\, \mathsf{IN}(\mathcal{K}'') \cap S(a,r)} \exists r.$ traversal-concept-i $(b, \mathsf{S}, k-1)$
 3: end if

Procedure traversal-concept-c ($A$, S, $k$)
Input: $A$: concept name from $\mathcal{K}''$; S: set of completion sets; $k \in \mathbb{N}$
Output: role-depth bounded traversal concept.
 1: if $k = 0$ then return $\bigsqcap_{B \in S(A)} B$
 2: else return $\bigsqcap_{B \in S(A)} B \sqcap \bigsqcap_{r \in \mathsf{RN}(\mathcal{K}'')} \bigsqcap_{B \in S(A,r)} \exists r.$traversal-concept-c $(B, \mathsf{S}, k-1)$
 3: end if
```

Lemma 2. *Let $\mathcal{K}$ be an $\mathcal{EL}$-KB, $\mathcal{K}''$ its normalized version, $\mathsf{S}_\mathcal{K}$ be the set of completion sets obtained for $\mathcal{K}$, $k \in \mathbb{N}$ a natural number and $a \in \mathsf{IN}(\mathcal{K})$. Furthermore let $C = k$-$\mathsf{C}_{\mathsf{S}_\mathcal{K}}(a)$ and $\widehat{C}$ be obtained from C by removing the normalization names. Then*

$$\mathcal{K}'' \models C(a) \textit{ iff } \mathcal{K} \models \widehat{C}(a).$$

This lemma guarantees that removing the normalization names from the traversal concept preserves the instance relationships. Intuitively, this lemma holds since the construction of the traversal concept conjoins exhaustively all named subsumers and all subsuming existential restrictions to a normalization name up to the role-depth bound. Thus removing the normalization name does not change the extension of the conjunction. The proof can be found in [18]. We are now ready to devise a computation algorithm for the role-depth bounded msc: procedure k-msc as displayed in Algorithm 2.

The procedure k-msc has an individual a from a knowledge base $\mathcal{K}$, the knowledge base $\mathcal{K}$ itself and number k for the role depth-bound as parameter. It first performs the two normalization steps on $\mathcal{K}$, then applies the completion rules from Figure 2 to the normalized KB $\mathcal{K}''$ and stores the set of completion sets in $\mathsf{S}_\mathcal{K}$. Afterwards it computes the traversal-concept of a from $\mathsf{S}_\mathcal{K}$ w.r.t. role-depth bound k. In a post-processing step it applies Remove-normalization-names to the traversal concept.

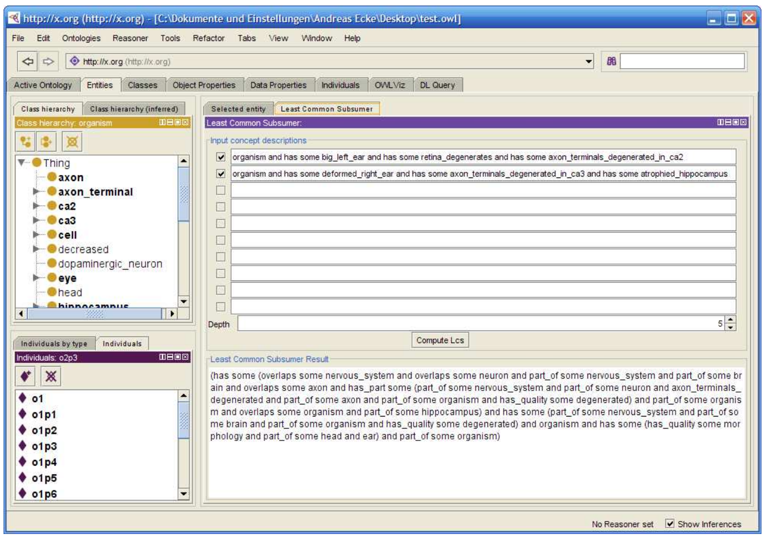
Obviously, the concept description returned from the procedure k-msc has a role-depth less or equal to k. The other conditions of Definition 4 are fulfilled as well, which has been shown in [18] yielding the correctness of the overall procedure.

Theorem 2. *Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be an $\mathcal{EL}$-KB and a an individual in $\mathcal{A}$ and $k \in \mathbb{N}$. Then* $\mathsf{k\text{-}msc}(a, \mathcal{K}, k) \equiv k\text{-}msc_{\mathcal{K}}(a)$.

The k-msc can grow exponential in the size of the knowledge base.

5 Implementation of GEL

The completion algorithm for classifying $\mathcal{EL}$ TBoxes was first implemented in the CEL reasoner [5]. We used its successor system JCEL [16] as a starting point for our implementation for the computation of the role-depth bounded lcs and msc. The implementation was done in Java and provides a simple GUI for the ontology editor PROTÉGÉ as can be seen in the screen-shot in Figure 3.


http://x.org (http://x.org) - [C:\Dokumente und Einstellungen\Andreas Ecke\Desktop\test.owl]
File Edit Ontologies Reasoner Tools Refactor Tabs View Window Help
Active Ontology Entities Classes Object Properties Data Properties Individuals OWLViz DL Query
Class hierarchy Class hierarchy (inferred)
Class hierarchy: organism
Thing
axon
axon_terminal
ca2
ca3
cell
decreased
dopaminergic_neuron
eye
head
hippocampus
Individuals by type Individuals
Individuals: o2p3
o1
o1p1
o1p2
o1p3
o1p4
o1p5
o1p6
Selected entity Least Common Subsumer
Least Common Subsumer:
Input concept descriptions
Least Common Subsumer Result
Compute Lcs
No Reasoner set
Show Inferences

☑ organism and has some big_left_ear and has some retina_degenerates and has some axon_terminals_degenerated_in_ca2

☑ organism and has some deformed_right_ear and has some axon_terminals_degenerated_in_ca3 and has some atrophied_hippocampus

Depth 5

```
(has some (overlaps some nervous_system and overlaps some neuron and part_of some nervous_system and part_of some br
ain and overlaps some axon and has_part some (part_of some nervous_system and part_of some neuron and axon_terminals_
degenerated and part_of some axon and part_of some organism and has_quality some degenerated) and part_of some organis
m and overlaps some organism and part_of some hippocampus) and has some (part_of some nervous_system and part_of so
me brain and part_of some organism and has_quality some degenerated) and organism and has some (has_quality some mor
phology and part_of some head and ear) and part_of some organism)
```

Fig. 3. LCS plugin

Our implementation of the methods presented here accesses the internal data structures of JCEL directly, providing a full integration of GEL into JCEL. The reasoning methods in GEL are in this first version realized in a naive way and are still in need of optimizations in order to handle the large knowledge bases that can be encountered in practice.

The concept descriptions returned by the lcs and the msc can grow exponentially in the worst case. On top of that, the returned concept descriptions are quite redundant in our current implementation, which might be acceptable if used as an input for a similarity measure, but surely not if presented to a human reader. It is future work to investigate methods for minimal rewritings of concept descriptions w.r.t. a general $\mathcal{EL}$ knowledge base in order to be able to present redundancy-free concept descriptions. Our tool will be made available as a plug-in for the ontology editor PROTÉGÉ and an API for the k-limited lcs and -msc is planned. The former system sonic [24] implemented the lcs and msc as well, but allowed only for acyclic, unfoldable TBoxes.

6 Conclusions

In this paper we have presented a practical method for computing the role-depth bounded lcs and the role-depth bounded msc of $\mathcal{EL}$-concepts w.r.t. a general TBox. We have argued that such generalization inferences are useful for ontology-based applications in many ways. Our approach for computing (approximations of) these inferences is based on the completion sets that are computed during classification of a TBox or realization of an ABox. Thus, any of the available implementations of the $\mathcal{EL}$ completion algorithm can be easily extended to an implementation of the two generalization inferences considered here. The same idea can be adapted for the computation of generalizations in the probabilistic DL Prob-$\mathcal{EL}_c^{01}$ [17, 18].

These theoretical results complete the (approximative) bottom-up approach for general $\mathcal{EL}$- (and Prob-$\mathcal{EL}_c^{01}$-) KBs. Continuing on the theoretical side, we want to investigate the bottom-up constructions (i.e. lcs and msc computations) in more expressive members of the $\mathcal{EL}$-family. We want to extend the approximative methods to $\mathcal{EL}$++, which extends $\mathcal{EL}$, for example, by transitive roles and role hierarchies. Such an extension would enable generalization reasoning services for the OWL 2 EL profile. Another interesting extension is to allow for more expressive means for probabilities.

Although a non-redundant representation of the concept descriptions obtained by the approximative lcs and msc is desirable when presented to a human reader, it is not clear whether a minimal representation of the obtained concept descriptions is favorable in every case. It might depend on the similarity measures employed whether a redundant representation of a concept is preferable over a compact one.

On the practical side, our future work will include evaluations of the usefulness of the offered reasoning services for biomedical applications and the development and testing of optimizations regarding the performance of the implementation.

Acknowledgments: We would like to thank Andreas Ecke and Julian Mendez for their implementation effort.

References

1. F. Baader. Least common subsumers and most specific concepts in a description logic with existential restrictions and terminological cycles. In G. Gottlob and T. Walsh, editors, *Proc. of the 18th Int. Joint Conf. on Artificial Intelligence (IJCAI-03)*, pages 325–330. Morgan Kaufmann, 2003.
2. F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ envelope. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI-05)*, Edinburgh, UK, 2005. Morgan-Kaufmann Publishers.

3. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.
4. F. Baader, R. Küsters, and R. Molitor. Computing least common subsumers in description logics with existential restrictions. In T. Dean, editor, *Proc. of the 16th Int. Joint Conf. on Artificial Intelligence (IJCAI-99)*, pages 96–101, Stockholm, Sweden, 1999. Morgan Kaufmann, Los Altos.
5. F. Baader, C. Lutz, and B. Suntisrivaraporn. CEL—a polynomial-time reasoner for life science ontologies. In U. Furbach and N. Shankar, editors, *Proc. of the 3rd Int. Joint Conf. on Automated Reasoning (IJCAR-06)*, volume 4130 of *Lecture Notes In Artificial Intelligence*, pages 287–291. Springer-Verlag, 2006. CEL download page: `http://lat.inf.tu-dresden.de/systems/cel/`.
6. A. Borgida, T. Walsh, and H. Hirsh. Towards measuring similarity in description logics. In *Proc. of the 2005 Description Logic Workshop (DL 2005)*, volume 147 of *CEUR Workshop Proceedings*, 2005.
7. W. W. Cohen, A. Borgida, and H. Hirsh. Computing least common subsumers in description logics. In W. Swartout, editor, *Proc. of the 10th Nat. Conf. on Artificial Intelligence (AAAI-92)*, pages 754–760, San Jose, CA, 1992. AAAI Press/The MIT Press.
8. S. Colucci, E. Di Sciascio, F. M. Donini, and E. Tinelli. Partial and informative common subsumers in description logics. In *proc. of 18th European Conference on Artificial Intelligence (ECAI 2008)*. IOS Press, 2008.
9. T. G. O. Consortium. Gene Ontology: Tool for the unification of biology. *Nature Genetics*, 25:25–29, 2000.
10. C. d'Amato, N. Fanizzi, and F. Esposito. A semantic similarity measure for expressive description logics. In *Pro.c of Convegno Italiano di Logica Computazionale, CILC05*, 2005.
11. M. C. Keet, M. Roos, and M. S. Marshall. A survey of requirements for automated reasoning services for bio-ontologies in OWL. In *Proceedings of Third international Workshop OWL: Experiences and Directions (OWLED 2007)*, 2007.
12. R. Küsters and R. Molitor. Approximating most specific concepts in description logics with existential restrictions. *AI Communications*, 15(1):47–59, 2002.
13. P. W. Lord, R. D. Stevens, A. Brass, and C. A. Goble. Investigating semantic similarity measures across the gene ontology: The relationship between sequence and annotation. *Bioinformatics*, 19(10):1275–1283, 2003.
14. C. Lutz, R. Piro, and F. Wolter. Enriching el-concepts with greatest fixpoints. In *Proc. of the 19th European Conf. on Artificial Intelligence (ECAI-10)*. IOS Press, 2010.
15. T. Mantay and R. Möller. Content-based information retrieval by computation of least common subsumers in a probabilistic description logic. In *Proc. International Workshop on Intelligent Information Integration, ECAI'98, Aug. 23-28, Brighton UK, 1998,*, 1998.
16. J. Mendez and B. Suntisrivaraporn. Reintroducing CEL as an OWL 2 EL reasoner. In B. Cuenca Grau, I. Horrocks, B. Motik, and U. Sattler, editors, *Proc. of the 2008 Description Logic Workshop (DL 2009)*, volume 477 of *CEUR-WS*, 2009.
17. R. Peñaloza and A.-Y. Turhan. Role-depth bounded least common subsumers by completion for $\mathcal{EL}$- and Prob-$\mathcal{EL}$-TBoxes. In V. Haarslev, D. Toman, and G. Weddell, editors, *Proc. of the 2010 Description Logic Workshop (DL'10)*, 2010.
18. R. Peñaloza and A.-Y. Turhan. Towards approximative most specific concepts by completion for $\mathcal{EL}^{01}$ with subjective probabilities. In T. Lukasiewicz, R. Peñaloza, and A.-Y. Turhan, editors, *Proceedings of the First International Workshop on Uncertainty in Description Logics (UniDL'10)*, 2010.
19. C. Pesquita, D. Faria, A. O. Falco, P. Lord, and F. M. Couto. Semantic similarity in biomedical ontologies. *PLoS Comput Biol*, 5, 2009.

20. P. Resnik. Using information content to evaluate semantic similarity in a taxonomy. In *Proc. of the 14th Int. Joint Conf. on Artificial Intelligence (IJCAI-95)*, pages 448–453, 1995.
21. K. Spackman. Managing clinical terminology hierarchies using algorithmic calculation of subsumption: Experience with snomed-rt. *Journal of the American Medical Informatics Assoc.*, 2000. Fall Symposium Special Issue.
22. T. Springer and A.-Y. Turhan. Employing description logics in ambient intelligence for modeling and reasoning about complex situations. *Journal of Ambient Intelligence and Smart Environments*, 1(3):235–259, 2009.
23. A.-Y. Turhan. *On the Computation of Common Subsumers in Description Logics*. PhD thesis, TU Dresden, Institute for Theoretical Computer Science, 2007.
24. A.-Y. Turhan and C. Kissig. SONIC — Non-standard inferences go OILED. In D. Basin and M. Rusinowitch, editors, *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR-04)*, volume 3097 of *Lecture Notes in Computer Science*, pages 321–325. Springer, 2004. SONIC is available from `http://wwwtcs.inf.tu-dresden.de/~sonic/`.
25. W3C OWL Working Group. OWL 2 web ontology language document overview. W3C Recommendation, 27th October 2009. `http://www.w3.org/TR/2009/REC-owl2-overview-20091027/`.