A Generic Approach for Correcting Access Restrictions to a Consequence

Martin Knechtel[1] and Rafael Peñaloza[2]

[1] SAP Research Center Dresden
martin.knechtel@sap.com
[2] Theoretical Computer Science TU Dresden, Germany
penaloza@tcs.inf.tu-dresden.de

Abstract. Recent research has shown that annotations are useful for representing access restrictions to the axioms of an ontology and their implicit consequences. Previous work focused on assigning a label, representing its access level, to each consequence from a given ontology. However, a security administrator might not be satisfied with the access level obtained through these methods. In this case, one is interested in finding which axioms would need to get their access restrictions modified in order to get the desired label for the consequence. In this paper we look at this problem and present algorithms for solving it with a variety of optimizations. We also present first experimental results on large scale ontologies, which show that our methods perform well in practice.

1 Introduction

In several applications it is desirable to have one, usually large, ontology, but offer different users access to different views of this ontology. In other words, each user has access to only a subset of the ontology, selected in accordance to an appropriate criterion. Different criteria can be used: access rights is only one of them, others are granularity, certainty, relevancy, trust etc., without loss of generality we focus on access rights in this paper while the results remain applicable to all the other lattice-based applications. Axioms have a privacy level and users get assigned a security clearance. A user can then only see those axioms whose privacy level is dominated by the security clearance of the user. In order to maintain only one ontology, we want to be able to store and retrieve the access information of users and axioms in an easy way. The approach proposed in [2] is to use a labeling lattice $(L, \leq)$; i. e. a set L of labels together with a partial order $\leq$ such that every finite set of labels has a join (least upper bound) and a meet (greatest lower bound) w.r.t. $\leq$. Every axiom a in the ontology $\mathcal{O}$ is assumed to have a label $\mathsf{lab}(a) \in L$, and each user receives also a label $\ell \in L$. The sub-ontology to which a user with label ℓ has access is defined as

$$\mathcal{O}_{\geq \ell} := \{a \in \mathcal{O} \mid \mathsf{lab}(a) \geq \ell\}.$$

A desirable property in ontologies is that they express as least explicit information as possible, while all the implicitly encoded knowledge is accessible through

L. Aroyo et al. (Eds.): ESWC 2010, Part I, LNCS 6088, pp. 167–182, 2010.

reasoning. Whenever we obtain a consequence c from an ontology, we need to be able to detect which users are allowed to see this consequence; in other words, if the user has a label ℓ, we need to decide whether c also follows from the sub-ontology $\mathcal{O}_{\geq \ell}$. The solution presented in [2] is to compute a so-called *boundary* for the consequence c; that is, an element ν_c of L such that c is a consequence of $\mathcal{O}_{\geq \ell}$ iff $\ell \leq \nu_c$.

The general problem can be seen similar to reducing inference control to access control in databases [8,7]. Inference control assumes a set of defined secrets and checks at runtime on every response to a user's query whether this response together with the user's a priori knowledge and already delivered answers implies any secret. In contrast to that, access control is enforced by following a policy which regulates access on explicit data. For knowledge bases, we extended access control to implied knowledge by computing a boundary for a consequence as described above. A security administrator can change the boundary only by changing the implying axioms' labels. For this purpose, he might be interested in support to find the minimal set of those axioms.

Just as ontology development and maintenance is an error prone activity, so is the adequate labeling of axioms according to their access requirements. A wrong access labeling may allow a user to deduce consequences for which he should have no security clearance. One may also encounter the dual problem, where a consequence is restricted to a user having the adequate security clearance. Both of these errors can be detected by looking at the boundary computed for the consequence. The first case occurs when the boundary is set too high up in the lattice, while the second case arises when the boundary is lower than expected.

The problem we want to solve then is that of repairing the labeling: we want to find a relabeling of the ontology such that the boundary computed under this new labeling yields the desired privacy level. This problem differs from that of ontology repair in that we do not aim to modify the axioms in the ontology, but only the labeling they receive. Our approach focuses on finding a minimal sub-ontology $\mathcal{S}$ such that, if we relabel all axioms in $\mathcal{S}$ to the goal label ℓ_g, and leave all other labels unchanged, then the boundary is changed to ℓ_g. Such a set will be called a *change set*. In order to commit as least changes as possible, we will prefer change sets of smaller cardinality.

We show that the ideas of axiom pinpointing [13,12,11,4,3] can be adapted to the search of a change set with minimum cardinality. We present black-box methods, improving upon the Hitting Set Tree approach to axiom pinpointing [10,14] that yield the desired change set. The methods take advantage of our search of a set with minimum cardinality, as well as the axiom labeling to reduce the search space and hence also the execution time. The experimental results at the end of the paper show that these enhancements improve the execution time.

2 Basic Notions and Results

To keep our presentation and results as general as possible, we impose only minimal restrictions to our ontology language. We just assume that an *ontology*

is a finite set, whose elements are called *axioms*, such that every subset of an ontology is itself an ontology. If $\mathcal{O}' \subseteq \mathcal{O}$ and $\mathcal{O}$ is an ontology, then $\mathcal{O}'$ is called a *sub-ontology* of $\mathcal{O}$. An ontology language specifies which sets of axioms are admitted as ontologies. Given an ontology language, a *monotone consequence relation* $\models$ is a binary relation between ontologies $\mathcal{O}$ and *consequences* c such that if $\mathcal{O} \models c$, then for every ontology $\mathcal{O}' \supseteq \mathcal{O}$ it holds that $\mathcal{O}' \models c$. If $\mathcal{O} \models c$, we say that c *follows from* $\mathcal{O}$ or that $\mathcal{O}$ *entails* c.

If $\mathcal{O} \models c$, we may be interested in finding the axioms responsible for this fact. A sub-ontology $\mathcal{S} \subseteq \mathcal{O}$ is called a *MinA* for $\mathcal{O}$,c if $\mathcal{S} \models c$ and for every $\mathcal{S}' \subset \mathcal{S}, \mathcal{S}' \not\models c$. The dual notion of a MinA is that of a diagnosis. A *diagnosis* for $\mathcal{O}$,c is a sub-ontology $\mathcal{S} \subseteq \mathcal{O}$ such that $\mathcal{O} \setminus \mathcal{S} \not\models c$ and $\mathcal{O} \setminus \mathcal{S}' \models c$ for all $\mathcal{S}' \subset \mathcal{S}$.

For a lattice $(L, \leq)$ and a set $K \subseteq L$, we denote as $\bigoplus_{\ell \in K} \ell$ and $\bigotimes_{\ell \in K} \ell$ the *join* (least upper bound) and *meet* (greatest lower bound) of K, respectively. We consider that ontologies are *labeled* with elements of the lattice. More formally, for an ontology $\mathcal{O}$ there is a labeling function lab that assigns a *label* $\mathsf{lab}(a) \in L$ to every element a of $\mathcal{O}$. We will often use the notation $L_{\mathsf{lab}} := \{\mathsf{lab}(a) \mid a \in \mathcal{O}\}$. For an element $\ell \in L$, we denote as $\mathcal{O}_{\geq \ell}$ the sub-ontology $\mathcal{O}_{\geq \ell} := \{a \in \mathcal{O} \mid \mathsf{lab}(a) \geq \ell\}$. The sub-ontologies $\mathcal{O}_{\leq \ell}, \mathcal{O}_{=\ell}, \mathcal{O}_{\neq \ell}, \mathcal{O}_{\not\geq \ell}$, and $\mathcal{O}_{\not\leq \ell}$ are defined analogously. This notion is also extended to sets of labels in the natural way, e.g. $\mathcal{O}_{=K} := \{a \in \mathcal{O} \mid \mathsf{lab}(a) = \ell \text{ for some } \ell \in K\}$. Conversely, for a sub-ontology $\mathcal{S} \subseteq \mathcal{O}$, we define $\lambda_{\mathcal{S}} := \bigotimes_{a \in \mathcal{S}} \mathsf{lab}(a)$ and $\mu_{\mathcal{S}} := \bigoplus_{a \in \mathcal{S}} \mathsf{lab}(a)$. An element $\ell \in L$ is called *join prime relative to* L_{lab} if for every $K_1, \ldots, K_n \subseteq L_{\mathsf{lab}}$, $\ell \leq \bigoplus_{i=1}^{n} \lambda_{K_i}$ implies that there is $i, 1 \leq i \leq n$ such that $\ell \leq \lambda_{K_i}$. Join prime elements relative to L_{lab} are called *user labels*. The set of all user labels is denoted as U. When dealing with labeled ontologies, the reasoning problem of interest consists on the computation of a boundary for a consequence c. Intuitively, the boundary divides the user labels ℓ of U according to whether $\mathcal{O}_{\geq \ell}$ entails c or not.

Definition 1 (Boundary). *Let $\mathcal{O}$ be an ontology and c a consequence. An element $\nu \in L$ is called a* boundary *for $\mathcal{O}$,c if for every join prime element relative to L_{lab} ℓ it holds that $\ell \leq \nu$ iff $\mathcal{O}_{\geq \ell} \models c$.*

Given a user label ℓ_u, we will say that the user *sees* a consequence c if $\ell_u \leq \nu$ for some boundary ν. The following lemma relating MinAs and boundaries was shown in [2].

Lemma 1. *If $\mathcal{S}_1, \ldots, \mathcal{S}_n$ are all MinAs for $\mathcal{O}$,c, then $\bigoplus_{i=1}^{n} \lambda_{\mathcal{S}_i}$ is a boundary for $\mathcal{O}$,c.*

A dual result, which relates the boundary with the set of diagnoses, also exists. The proof follows easily from the definitions given in this section.

Lemma 2. *If $\mathcal{S}_1, \ldots, \mathcal{S}_n$ are all diagnoses for $\mathcal{O}$,c, then $\bigotimes_{i=1}^{n} \mu_{\mathcal{S}_i}$ is a boundary for $\mathcal{O}$,c.*

Example 1. Let $(L_d, \leq_d)$ be the lattice shown in Figure 1, and $\mathcal{O}$ a labeled ontology from a marketplace in the Semantic Web with the following axioms

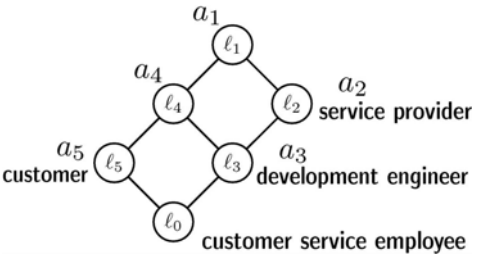

Fig. 1. Lattice $(L_d, \leq_d)$ with 4 user labels and an assignment of 5 axioms to labels

$a_1 : EUecoService \sqcap HighperformanceService(ecoCalculatorV1)$
$a_2 : HighperformanceService$
$\quad \sqsubseteq ServiceWithLowCustomerNr \sqcap LowProfitService$
$a_3 : EUecoService \sqsubseteq ServiceWithLowCustomerNr \sqcap LowProfitService$
$a_4 : ServiceWithLowCustomerNr \sqsubseteq ServiceWithComingPriceIncrease$
$a_5 : LowProfitService \sqsubseteq ServiceWithComingPriceIncrease$

where the function lab assigns to each axiom the labels as shown in Figure 1. This ontology entails $c : ServiceWithComingPriceIncrease(ecoCalculatorV1)$. The MinAs for $\mathcal{O}$,c are $\{a_1, a_2, a_4\}, \{a_1, a_2, a_5\}, \{a_1, a_3, a_4\}, \{a_1, a_3, a_5\}$, and its diagnoses are $\{a_1\}, \{a_2, a_3\}, \{a_4, a_5\}$. Using Lemma 2, we can compute the boundary as $\mu_{\{a_1\}} \otimes \mu_{\{a_2,a_3\}} \otimes \mu_{\{a_4,a_5\}} = \ell_1 \otimes \ell_2 \otimes \ell_4 = \ell_3$. Valid user labels are $\ell_0, \ell_2, \ell_3, \ell_5$ which represent user roles as illustrated. For ℓ_0 and ℓ_3, c is visible.

3 Modifying the Boundary

Once the boundary for a consequence c has been computed, it is possible that the knowledge engineer or the security administrator considers this solution erroneous. For instance, the boundary may express that a given user u is able to deduce c, although this was not intended. Alternatively, the boundary may imply that c is a confidential consequence, only visible to a few, high-clearance users, while in reality c should be publicly available. The problem we face is how to change the labeling function so that the computed boundary corresponds to the desired label in the lattice. This problem can be formalized and approached in several different ways. In our approach, we fix a goal label ℓ_g and try to modify the labeling of as least axioms as possible so that the boundary equals ℓ_g.

Definition 2. *Let $\mathcal{O}$ be an ontology, c a consequence,* lab *a labeling function, $\mathcal{S} \subseteq \mathcal{O}$ and $\ell_g \in L$ the* goal label. *The* modified assignment $\mathsf{lab}_{\mathcal{S},\ell_g}$ *is given by*

$$\mathsf{lab}_{\mathcal{S},\ell_g}(t) = \begin{cases} \ell_g, & \text{if } t \in \mathcal{S}, \\ \mathsf{lab}(t), & \text{otherwise.} \end{cases}$$

A sub-ontology $\mathcal{S} \subseteq \mathcal{O}$ is called a change set *for ℓ_g if the boundary for $\mathcal{O}$,c under the labeling function $\mathsf{lab}_{\mathcal{S},\ell_g}$ equals ℓ_g.*

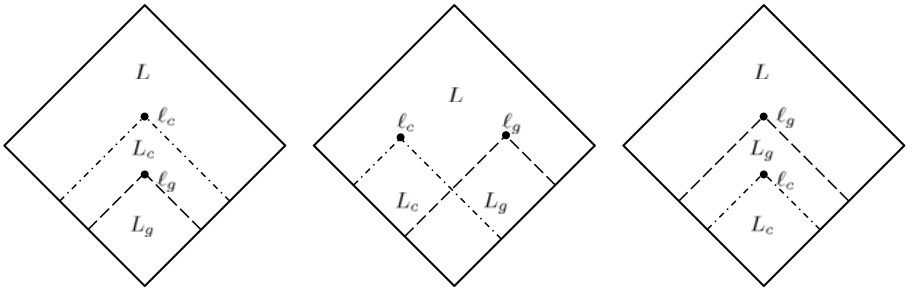

Fig. 2. Hide consequence from some user roles (left), allow additional user roles to see consequence (right) and both at the same time (middle)

Obviously, the original ontology $\mathcal{O}$ is always a change set for any goal label if $\mathcal{O} \models c$. However, we are interested in performing minimal changes to the labeling function. Hence, we search for a change set of minimum cardinality. It follows from [5] that this problem is NP-complete.

Let ℓ_g denote the goal label and ℓ_c the computed boundary for c. There are three possible cases for ℓ_g and ℓ_c to differ, which are illustrated in Figure 2: either (i) $\ell_g < \ell_c$ (left), (ii) $\ell_c < \ell_g$ (right), or (iii) ℓ_g and ℓ_c are incomparable (middle). The sets L_c and L_g contain the user labels before and after the label changes respectively. Consider the first case, where $\ell_g < \ell_c$. Then, from Lemma 2 it follows that any diagnosis $\mathcal{S}$ is a change set for ℓ_g: since $\ell_g < \ell_c$, then for every diagnosis $\mathcal{S}'$, $\ell_g < \mu_{\mathcal{S}'}$. But then, under the new labeling $\mathsf{lab}_{\mathcal{S},\ell_g}$ we get that $\mu_{\mathcal{S}} = \ell_g$. And hence, when the greatest lower bound of all $\mu_{\mathcal{S}'}$ is computed, we obtain ℓ_g as a boundary. Using an analogous argument and Lemma 1, it is possible to show that if $\ell_c < \ell_g$, then every MinA is a change set for ℓ_g. The third case can be solved using a combination of the previous two: if ℓ_g and ℓ_c are incomparable, we can first set as a partial goal $\ell'_g := \ell_g \otimes \ell_c$. Thus, we can first solve the first case, to set the boundary to ℓ'_g, and then, using the second approach, modify this new boundary once more to ℓ_g. Unfortunately, this approach does not yield a change set of minimum cardinality, even if the smallest diagnosis or MinA is computed, as shown in the following example.

Example 2. Let $\mathcal{O}$,c and lab be as in Example 1. We then know that $\ell_c := \ell_3$ is a boundary for $\mathcal{O}$,c. Suppose now that the goal label is $\ell_g := \ell_4$. Since $\ell_c < \ell_g$, we know that any MinA is a change set. Since all MinAs for $\mathcal{O}$,c have exactly three elements, any change set produced this way will have cardinality three. However, $\{a_2\}$ is also a change set, whose cardinality is obviously smaller.

To understand why the minimality of MinAs is not sufficient for obtaining a minimum change set, we can look back to Lemma 1. This lemma says that in order to find a boundary, we need to compute the join of all $\lambda_{\mathcal{S}}$, with $\mathcal{S}$ a MinA, and $\lambda_{\mathcal{S}}$ the meet of the labels of all axioms in $\mathcal{S}$. But then, for any axiom $a \in \mathcal{S}$ such that $\ell_g \leq \mathsf{lab}(a)$, modifying this label to ℓ_g will have no influence in the result of $\lambda_{\mathcal{S}}$. In Example 2, there is a MinA $\{a_1, a_2, a_4\}$, where two axioms, namely a_1 and a_4 have a label greater or equal to $\ell_g = \ell_4$. Thus, the only axiom

that needs to be relabeled is in fact a_2, which yields the minimum change set $\{a_2\}$ shown in the example. Basically, we consider every axiom $a \in \mathcal{O}$ such that $\ell_g \leq \mathsf{lab}(a)$ as *fixed* in the sense that it is superfluous for any change set. For this reason, we will deal with a generalization of MinAs and diagnoses, that we call IAS and RAS, respectively.

Definition 3 (IAS,RAS). *A* minimal inserted axiom set *(IAS) for ℓ_g is a subset $I \subseteq \mathcal{O}_{\not\geq \ell_g}$ such that $\mathcal{O}_{\geq \ell_g} \cup I \models c$ and for every $I' \subset I : \mathcal{O}_{\geq \ell_g} \cup I' \not\models c$.*

A minimal removed axiom set *(RAS) for ℓ_g is a subset $R \subseteq \mathcal{O}_{\not\leq \ell_g}$ such that $\mathcal{O}_{\not\leq \ell_g} \setminus R \not\models c$ and for every $R' \subset R : \mathcal{O}_{\not\leq \ell_g} \setminus R' \models c$.*

The following theorem justifies the use of IAS and RAS when searching for change sets of minimum cardinality.

Theorem 1. *Let ℓ_c be a boundary for $\mathcal{O}$,c, ℓ_g the goal label, I an IAS for ℓ_g of minimum cardinality and R an RAS for ℓ_g of minimum cardinality. Then, the following holds:*

- *if $\ell_g < \ell_c$, then R is a change set of minimum cardinality,*
- *if $\ell_c < \ell_g$, then I is a change set of minimum cardinality.*

4 Computing the Smallest IAS and RAS

In this section, we show how the smallest IAS and RAS can be computed. We first present the most obvious approach that is based in the computation of all MinAs and diagnoses. Afterwards, we show how this idea can be improved by considering fixed portions of the ontology, as described before. We further improve this approach by showing that it usually suffices to compute only partial IAS and RAS, thus reducing the search space and execution time of our method.

4.1 Using Full Axiom Pinpointing

Although we have shown in Example 2 that MinAs and diagnoses do not yield change sets of minimum cardinality directly, these change sets can still be deduced from the set of all MinAs and diagnoses, as shown by the following lemma.

Lemma 3. *Let I (R) be an IAS (RAS) for ℓ_g, then there is a MinA (diagnosis) $\mathcal{S}$ such that $I = \mathcal{S} \setminus \mathcal{O}_{\geq \ell_g}$ ($R = \mathcal{S} \setminus \mathcal{O}_{\leq \ell_g}$).*

Lemma 3 shows that we can compute the set of all IAS by first computing all MinAs and then removing the set of fixed elements $\mathcal{O}_{\geq \ell_g}$ from it[1]. Thus, the most naïve approach for computing a change set of minimum cardinality is to first find all MinAs, then compute the set of all IAS by removing all elements in $\mathcal{O}_{\geq \ell_g}$, and finally search for the IAS having the least elements.

[1] To avoid unnecessary repetitions, we henceforth focus our discussion on the computation of the smallest IAS except for very specific cases. It should be noted, however, that the case of RAS can be treated in a similar fashion.

The task of computing all MinAs, also called axiom pinpointing, has been widely studied in recent years, and there exist black-box implementations based on the hitting set tree (HST) algorithm [10,14]. The HST algorithm makes repeated calls to an auxiliary procedure that computes a single MinA. Further MinAs are found by building a tree, where nodes are labeled with MinAs. If the MinA labeling a node has n axioms ($\mathcal{S} := \{a_1, \ldots, a_n\}$), then this node will have n children: the i-th child is labeled with a MinA obtained after removing a_i from the ontology. This ensures that each node is labeled with a MinA distinct from those of its predecessors. Although not stated explicitly in the axiom pinpointing literature, it is clear that the same HST algorithm can be used for computing all diagnoses. The only variant necessary is to have a subroutine capable of computing one such diagnosis, which can be obtained by dualizing the algorithm computing one MinA (see Algorithm 2 for an example on how this dualization works). In our experiments, we used this approach as a basis to measure the improvement achieved by the optimizations that will be introduced next.

4.2 Using Fixed Sub-ontologies and Cardinality Limit

Rather than computing the set of IAS indirectly by first computing MinAs and then removing the fixed axioms, we would like to use a procedure that finds them directly. Moreover, since our goal is to find the *smallest* IAS, we want this procedure to stop once the IAS being computed is ensured to have cardinality larger than or equal to that of the best IAS found so far. In Algorithm 1 we present a variation of the logarithmic MinA extraction procedure presented in [6] that is able to compute an IAS or stop once this has reached a size n.

We also show the RAS variant in Algorithm 2 to illustrate how the duality between IAS and RAS transfers to the algorithms that compute them.

Given a goal label ℓ_g, if we want to compute an IAS of size at most n for a consequence c, then we would make a call to extract-partial-IAS$(\mathcal{O}_{\geq \ell_g}, \mathcal{O} \setminus \mathcal{O}_{\geq \ell_g}, c, n)$. Similarly, a call to extract-partial-RAS$(\mathcal{O} \setminus \mathcal{O}_{\leq \ell_g}, \mathcal{O} \setminus \mathcal{O}_{\leq \ell_g}, c, n)$ yields an RAS of size at most n. The second parameter defines the axioms which are allowed to be contained in the RAS. One of the advantages of the HST algorithm is that the labels of any node are always ensured not to contain the label of any of its predecessor nodes. In particular this means that even if we do not always compute a full IAS (like in Algorithm 1, where the output may be only a subset of size n of an IAS), the algorithm will still correctly find all IAS that do not contain any of the partial IAS found during the execution. Since we are interested in finding the IAS of minimum cardinality, we can set the limit n to the size of the smallest IAS found so far. If extract-partial-IAS outputs a set with fewer elements, we are sure that this is indeed a full IAS, and hence we can update our best result to this newly found set. The HST algorithm will not find all IAS in this way, but we can be sure that one IAS with the minimum cardinality will be found.

The idea of using only partial information for finding the smallest IAS can be taken a step further. We can in fact modify the HST algorithm itself, so that it looks only at a reduced search space, while still finding a smallest IAS. Algorithm 3 shows the modified HST method.

Algorithm 1. Compute (partial) IAS

Procedure extract-partial-IAS($\mathcal{O}_{\text{fix}}, \mathcal{O}_{\text{test}}, c, n$)
Input: $\mathcal{O}_{\text{fix}}$: fixed axioms; $\mathcal{O}_{\text{test}}$: axioms; c: consequence; n: limit
Output: first n elements of a minimal $\mathcal{S} \subseteq \mathcal{O}_{\text{test}}$ such that $\mathcal{O}_{\text{fix}} \cup \mathcal{S} \models c$

1: **Global** $l := 0, n$
2: **return** extract-partial-IAS-r($\mathcal{O}_{\text{fix}}, \mathcal{O}_{\text{test}}, c$)

Subprocedure extract-partial-IAS-r($\mathcal{O}_{\text{fix}}, \mathcal{O}_{\text{test}}, c$)

1: **if** $n = l$ **then**
2: **return** $\emptyset$
3: **if** $|\mathcal{O}_{\text{test}}| = 1$ **then**
4: $l := l + 1$
5: **return** $\mathcal{O}_{\text{test}}$
6: $\mathcal{S}_1, \mathcal{S}_2 :=$ halve($\mathcal{O}_{\text{test}}$) (partition $\mathcal{O}_{\text{test}}$ so that $||\mathcal{S}_1| - |\mathcal{S}_2|| \leq 1$)
7: **if** $\mathcal{O}_{\text{fix}} \cup \mathcal{S}_1 \models c$ **then**
8: **return** extract-partial-IAS-r($\mathcal{O}_{\text{fix}}, \mathcal{S}_1, c$)
9: **if** $\mathcal{O}_{\text{fix}} \cup \mathcal{S}_2 \models c$ **then**
10: **return** extract-partial-IAS-r($\mathcal{O}_{\text{fix}}, \mathcal{S}_2, c$)
11: $\mathcal{S}'_1 :=$ extract-partial-IAS-r($\mathcal{O}_{\text{fix}} \cup \mathcal{S}_2, \mathcal{S}_1, c$)
12: $\mathcal{S}'_2 :=$ extract-partial-IAS-r($\mathcal{O}_{\text{fix}} \cup \mathcal{S}'_1, \mathcal{S}_2, c$)
13: **return** $\mathcal{S}'_1 \cup \mathcal{S}'_2$

There are two main differences between the original HST method as described in [10,14] and the procedure hst-extract-smallest-IAS. The first one, as already explained before, is that the nodes of the generated tree are not necessarily labeled with an IAS, but may contain only a partial IAS. The reduction in the search space here is clear: since nodes have less axioms, the search tree has a lower branching factor. Moreover, the set of (possibly partial) IAS still to be found is also reduced, since no superset of any label set found so far is allowed. The second difference is that the tree is not expanded at each node for all the axioms in it, but rather only on the first $m - 1$, where m is the size of the smallest IAS found so far. This further reduces the search space by decreasing the branching factor of the search tree. Notice that the highest advantage of this second optimization appears when the HST is constructed in a depth-first fashion. In that case, a smaller IAS found further below in the tree will reduce the branching factor of all its predecessors. The following theorem shows that Algorithm 3 is correct.

Theorem 2. *Let $\mathcal{O}$ be an ontology, c a consequence with $\mathcal{O} \models c$, and ℓ_g a goal label. If n is the minimum cardinality of an IAS for ℓ_g, then Algorithm 3 outputs an IAS I such that $|I| = n$.*

Proof. Algorithm 3 always outputs an IAS since I is only modified when the output of extract-partial-IAS has size strictly smaller than the limit m, and hence only when this is an IAS itself. Suppose now that the output I is such that $n < |I|$, and let I_0 be an IAS such that $|I_0| = n$, which exists by assumption. Then, every set obtained by calls to extract-partial-IAS has size strictly greater

Algorithm 2. Compute (partial) RAS

Procedure extract-partial-RAS($\mathcal{O}_{\text{nonfix}}, \mathcal{O}_{\text{test}}, c, n$)
Input: $\mathcal{O}_{\text{nonfix}}$: axioms; $\mathcal{O}_{\text{test}}$: axioms; c: consequence; n: limit
Output: first n elements of a minimal $\mathcal{S} \subseteq \mathcal{O}_{\text{test}}$ such that $\mathcal{O}_{\text{nonfix}} \setminus \mathcal{S} \not\models c$

1: **Global** $l := 0, \mathcal{O} := \mathcal{O}_{\text{nonfix}}, n$
2: **return** extract-partial-RAS-r($\emptyset, \mathcal{O}_{\text{test}}, c$)

Subprocedure extract-partial-RAS-r($\mathcal{O}_{\text{fix}}, \mathcal{O}_{\text{test}}, c$)

1: **if** $n = l$ **then**
2: **return** $\emptyset$
3: **if** $|\mathcal{O}_{\text{test}}| = 1$ **then**
4: $l := l + 1$
5: **return** $\mathcal{O}_{\text{test}}$
6: $\mathcal{S}_1, \mathcal{S}_2 :=$ halve($\mathcal{O}_{\text{test}}$)
7: **if** $\mathcal{O} \setminus (\mathcal{O}_{\text{fix}} \cup \mathcal{S}_1) \not\models c$ **then**
8: **return** extract-partial-RAS-r($\mathcal{O}_{\text{fix}}, \mathcal{S}_1, c$)
9: **if** $\mathcal{O} \setminus (\mathcal{O}_{\text{fix}} \cup \mathcal{S}_2) \not\models c$ **then**
10: **return** extract-partial-RAS-r($\mathcal{O}_{\text{fix}}, \mathcal{S}_2, c$)
11: $\mathcal{S}'_1 :=$ extract-partial-RAS-r($\mathcal{O}_{\text{fix}} \cup \mathcal{S}_2, \mathcal{S}_1, c$)
12: $\mathcal{S}'_2 :=$ extract-partial-RAS-r($\mathcal{O}_{\text{fix}} \cup \mathcal{S}'_1, \mathcal{S}_2, c$)
13: **return** $\mathcal{S}'_1 \cup \mathcal{S}'_2$

than n, since otherwise, I and m would be updated. Consider now an arbitrary set $\mathcal{S}$ found during the execution through a call to extract-partial-IAS, and let $\mathcal{S}_n := \{a_1, \ldots, a_n\}$ the first n elements of $\mathcal{S}$. Since $\mathcal{S}$ is a (partial) IAS, it must follow that $I_0 \not\subseteq \mathcal{S}_n$. Then, there must be an $i, 1 \leq i \leq n$ such that $a_i \notin I_0$. But then, I_0 will still be an IAS after axiom $\{a_i\}$ has been removed. Since this argument is true for all nodes, it in particular holds at all leaf nodes, but then they cannot be leaf nodes, since a new IAS, namely I_0 can still be found by expanding the HST. This contradicts that I is the output of the algorithm. □

Efficient implementations of the original version of the HST algorithm rely on several optimizations. Two standard optimizations described in the literature are node-reuse and early path termination (see, e.g. [10,14,2]). Node-reuse keeps a history of all nodes computed so far in order to avoid useless (and usually expensive) calls to the auxiliary procedure that computes a new node. Early path termination, on the other hand, prunes the hitting set tree by avoiding expanding nodes when no new information can be derived from further expansion. In order to avoid unnecessary confusion, we have described the modified HST algorithm without including these optimizations. However, it should be clear that both, node-reuse and early path termination, can be included in the algorithm without destroying its correctness. All the implementations used in the experimental results contain these two optimizations.

4.3 Label-Based Optimization

Up to now, our approach has only looked at the axioms individually in an attempt to compute the IAS, but the labels of these axioms have been ignored,

Algorithm 3. Modified HST algorithm to find smallest IAS for ℓ_g

Procedure hst-extract-smallest-IAS$(\mathcal{O}, (L, \leq), c, \ell_g)$
Input: $\mathcal{O}$: labeled ontology; $(L, \leq)$: lattice; c: consequence of $\mathcal{O}$; ℓ_g: goal label
Output: IAS I of minimum cardinality

Global $I := \mathcal{O}, m := |\mathcal{O}|, \mathcal{O}_{\text{fix}} := \mathcal{O}_{\geq \ell_g}, \mathcal{O}_{\text{nonfix}} := \mathcal{O} \setminus \mathcal{O}_{\text{fix}}$
expand-hst-IAS$(\mathcal{O} \setminus \mathcal{O}_{\text{fix}}, c)$
return I

Procedure expand-hst-IAS$(\mathcal{O}, c)$
Input: $\mathcal{O}$: ontology; c: consequence
Side effects: modifications to I and m

if $\mathcal{O}_{\text{fix}} \cup \mathcal{O} \not\models c$ **then**
  return
else
  $\mathcal{S}$:= extract-partial-IAS$(\mathcal{O}_{\text{fix}}, \mathcal{O}, c, m)$
  if $|\mathcal{S}| < |I|$ **then**
    $I := \mathcal{S}$
    $m := |I|$
for the first $(m-1)$ axioms t in $\mathcal{S}$ **do**
  expand-hst-IAS$(\mathcal{O} \setminus \{t\}, c)$

except for the fixed axioms, whose label is greater than or equal to ℓ_g. The experiments in [2] show that it is usually faster to compute the set of labels that contain a MinA than a MinA itself. This is not surprising since usually a single lattice element ℓ labels more than one axiom, and hence when testing whether there is an axiom labeled with ℓ that belongs to a MinA we are simultaneously performing this test for several axioms. Intuitively, the same should be true for the computation of IAS. Hence, our search for the smallest IAS can be further improved if IAS are computed through a two-step procedure: (1) without computing the IAS we compute the set of labels K of axioms in the IAS, (2) we then find the IAS from only the axioms that have labels in K. Notice that for a given set of labels K, there can in fact be several IAS having labels in K. We can use a HST algorithm to find all of them before trying to compute a new label set.

Interestingly, this approach can also be improved when searching for the smallest IAS by using (partial) label sets of length at most that of the smallest IAS found so far. Algorithm 4 shows a simple procedure for computing such partial label sets, up to a given length n. Let L_{IAS} be the set obtained by an application of partial-lab-IAS$(\mathcal{O}, (L, \leq), L_{\geq \ell_g}, c, n)$, where $L_{\geq \ell_g} := \{\ell \in L \mid \ell \geq \ell_g\}$. Then, every IAS contained in $\mathcal{O}_{L_{\text{IAS}}}$ has at least one axiom labeled with each element ℓ of L_{IAS}; otherwise, ℓ would not have been added to the set during the application of the algorithm. That in particular means that for every IAS I contained in $\mathcal{O}_{L_{\text{IAS}}}$ it holds that $|L_{\text{IAS}}| \leq |I|$. Thus, there is no risk on stopping the computation of the label set when it is large enough, if we are only searching for the smallest IAS. Moreover, any IAS that can be deduced from this set of labels will not give any new information, so we can further avoid the second step that extracts specific IAS from the set of labels. Algorithm 5 shows how the two

Algorithm 4. Compute labels of (partial) IAS

Procedure partial-lab-IAS$(\mathcal{O}, (L, \leq), L_{\text{fix}}, c, n)$
Input: $\mathcal{O}$ ontology; $(L, \leq)$ lattice; L_{fix}: fixed labels; c: consequence; n: limit
Output: $L_{\text{IAS}} \subseteq L$: set of labels of at least one (partial) IAS of size at most n

```
1: S := O; L_IAS := ∅
2: for every k ∈ (L \ L_fix) do
3:    if S \ O_=k ⊨ c then
4:       S := S \ O_=k
5:    else
6:       L_IAS := L_IAS ∪ {k}
7:       if |L_IAS| = n then
8:          return L_IAS
9: return L_IAS
```

step computation of IAS can be combined with the modified HST approach of Algorithm 3 to efficiently obtain the smallest IAS.

The algorithm works as follows. It fist computes a set of labels of an IAS L_{IAS}. Using the restricted ontology having only axioms labeled with elements of L_{IAS}, it starts then the modified hitting set tree algorithm trying to find an IAS of smaller size. However, when the hitting set tree cannot be expanded further due to the fact that no other IAS appear in the ontology in use, the algorithm does not stop, but rather goes back to compute a new label set. Intuitively, is a nesting of two HST methods. The outer one only computes label sets in the style of [2], and the inner one uses restricted sub-ontologies based on the results of the outer one to compute the actual IAS. As said before, the method also uses the optimization in which the sets used in each of the hitting set trees are restricted in size, according to the smallest IAS found so far. Additionally, the inner HST is not called whenever the label set found by the outer procedure is already larger than the smallest IAS known, since no IAS found from that sub-ontology will be smaller, and hence is irrelevant in our search for the IAS with least axioms. The following theorem, stating the correctness of Algorithm 5, can be shown following an argument similar to the proof of Theorem 2.

Theorem 3. *Let $\mathcal{O}$ be an ontology, c a consequence with $\mathcal{O} \models c$, and ℓ_g a goal label. If n is the minimum cardinality of an IAS for ℓ_g, then Algorithm 5 outputs an IAS I such that $|I| = n$.*

5 Empirical Evaluation

We implemented and evaluated our algorithms empirically with large practical ontologies. The following sections describe our test setting and the results.

5.1 Test Data and Test Environment

We test on a PC with 2GB RAM and Intel Core Duo CPU 3.16GHz. We implemented all approaches with Java 1.6, CEL 1.0, Pellet 2.0 and OWL API trunk

Algorithm 5. Label optimized HST algorithm to find smallest IAS

Procedure lab-hst-extract-smallest-IAS$(\mathcal{O}, (L, \leq), c, \ell_g)$
Input: $\mathcal{O}$: labeled ontology; $(L, \leq)$: lattice; c: consequence; ℓ_g: goal boundary
Output: IAS I of minimum cardinality

1: **Global** $I := \mathcal{O}, m := |\mathcal{O}|, L_{\text{fix}} := \{\ell \in L \mid \ell \geq \ell_g\}, \mathcal{O}_{\text{fix}} := \mathcal{O}_{\geq \ell_g}, (L, \leq)$, $\mathcal{O}_{\text{nonfix}} := \mathcal{O} \setminus \mathcal{O}_{\text{fix}}$
2: expand-lab-hst-IAS$(\mathcal{O} \setminus \mathcal{O}_{\text{fix}}, c)$
3: **return** I

Procedure expand-lab-hst-IAS$(\mathcal{O}, c)$
Input: $\mathcal{O}$: ontology; c: consequence
Side effects: modifications to I and m

1: **if** $\mathcal{O}_{\text{fix}} \cup \mathcal{O} \not\models c$ **then**
2: **return**
3: $M :=$ partial-lab-IAS$(\mathcal{O}, (L, \leq), L_{\text{fix}}, c, m)$
4: **if** $|M| < m$ **then**
5: expand-hst-IAS-aux$(\mathcal{O}, \mathcal{O}_{=M}, c)$
6: **else**
7: **for** the first $m - 1$ labels $\ell \in M$ **do**
8: expand-lab-hst-IAS$(\mathcal{O}_{\neq \ell}, c)$

Procedure expand-hst-IAS-aux$(\mathcal{O}, \mathcal{O}_{\text{test}}, c)$
Input: $\mathcal{O}$: ontology; $\mathcal{O}_{\text{test}}$: axioms; c: consequence
Side effects: modifications to I and m

1: **if** $\mathcal{O}_{\text{fix}} \cup \mathcal{O}_{\text{test}} \not\models c$ **then**
2: expand-lab-hst-IAS$(\mathcal{O}, c)$
3: **else**
4: $\mathcal{S} :=$ extract-partial-IAS$(\mathcal{O}_{\text{fix}}, \mathcal{O}_{\text{test}}, c, m)$
5: **if** $|\mathcal{S}| < m$ **then**
6: $I := \mathcal{S}$
7: $m := |I|$
8: **for** the first $(m - 1)$ axioms t in $\mathcal{S}$ **do**
9: expand-hst-IAS-aux$(\mathcal{O} \setminus \{t\}, \mathcal{O}_{\text{test}} \setminus \{t\}, c)$

revision 1150. Since we need lattices, labeled ontologies and computed boundaries of their consequences, the test data used in [2] was feasible also in this context and we will describe the test data here only briefly. The two labeling lattices are similar to ones encountered in real-world applications: the nonlinear lattice $(L_d, \leq_d)$ was already introduced in Figure 1, the linear order $(L_l, \leq_l)$ has 6 elements $L_l = L_d = \{\ell_0, \ldots, \ell_5\}$ with $\leq_l := \{(\ell_n, \ell_{n+1}) \mid \ell_n, \ell_{n+1} \in L_l \wedge 0 \leq n \leq 5\}$.

We used the two ontologies O^{SNOMED} and O^{FUNCT} with different expressivity and types of consequences for our experiments. The Systematized Nomenclature of Medicine, Clinical Terms (SNOMED CT) is a comprehensive medical and clinical ontology built using the Description Logic (DL) $\mathcal{EL}^+$. From the January/2005 release of the DL version, which contains 379,691 concept names, 62 object property names, and 379,704 axioms, and entails more than five million subsumptions, we used a sampled set of 27,477 positive subsumptions. Following [6], for each subsumee A of any subsumption $A \sqsubseteq B$ we precomputed the

reachability-based module with CEL and stored these modules. This module is guaranteed to contain all axioms of any MinA for $A \sqsubseteq B$, so it can be used as the start ontology when searching for MinA, thus also for diagnoses, IAS and RAS rather than searching the complete ontology. O^{FUNCT} is an OWL-DL ontology for functional description of mechanical engineering solutions [9]. It has 115 concept names, 47 object property names, 16 data property names, 545 individual names, 3,176 axioms, and the DL expressivity is $\mathcal{SHOIN}(\mathbf{D})$. Its 716 consequences are 12 subsumption and 704 instance relationships (class assertions).

To obtain labeled ontologies, axioms in both labeled ontologies received a random label assignment of elements from $L_l = L_d$. As black-box subsumption and instance reasoner we used the reasoner Pellet since it can deal with the expressivity of both ontologies. For the expressive DL $\mathcal{SHOIN}(\mathbf{D})$ it uses a tableau-based algorithm and for $\mathcal{EL}+$ it uses an optimized classifier for the OWL 2 EL profile, which is based on the algorithm described in [1].

We computed the boundary ℓ_c of each consequence c of the ontologies with the algorithms described in [2] and then computed the change set described in this paper to reach goal boundary ℓ_g which is constantly ℓ_3 in all experiments. Consequences where $\ell_c = \ell_g$ were not considered. Thus, from the 716 consequences in O^{FUNCT}, we have 415 remaining with labeling lattice $(L_d, \leq_d)$ and 474 remaining with $(L_l, \leq_l)$. From the 27,477 consequences in O^{SNOMED} we have 23,695 remaining with labeling lattice $(L_d, \leq_d)$ and 25,897 with $(L_l, \leq_l)$.

5.2 Results

Table 1 contains results for the 4 combinations of the 2 ontologies and the 2 labeling lattices. For each of them we tested 4 variants, leading to 16 measurement series overall. We tested the variants *full axiom pinpointing* limited to 10 MinA, *fixed axioms*, *fixed axioms and cardinality limit* and *fixed axioms and cardinality limit and label-based optimization* all of the last three limited to 10 IAS and 10 RAS. Running with fixed axioms without cardinality limit can be done easily by skipping Line 7 in Algorithm 3.

Since we limit the number of MinAs, IAS and RAS, our algorithms might not find the smallest change set before reaching the limit. We measure the quality of the presented variants given those limitations at execution time. Table 1 lists the ratio of correct solutions where at least 1 correct change set was computed, and the ratio of optimal solutions where the limit was not reached during the computation and thus yielded the shortest change set possible. Notice however that the ratio of cases with the minimal change set successfully computed might be higher, including those where the limitation was reached but the minimal change set was already found.

Full axiom pinpointing is clearly outperformed by the other variants. Remarkably, label-based optimization seems not to pay off in our setting. A reason might be the small subset of the ontology, on which the Hitting Set Tree algorithms is working on already by fixing axioms. Furthermore in O^{SNOMED} we work on the

Table 1. Results of the optimizations in 4 test settings

Ont.	Lat.	Variant and Runtime Limit	Time (minutes)	Ratio of correct solutions	Ratio of optimal solutions
O^{FUNCT}	nonlinear	full PP, ≤ 10 MinA	44.05	96%	47%
		fixed axioms, ≤ 10 IAS/RAS	20.36	100%	91%
		fixed axioms, partial, ≤ 10 IAS/RAS	7.11	100%	99%
		fixed axioms, partial, lab-opt., ≤ 10 IAS/RAS	7.85	100%	99%
	linear	full PP, ≤ 10 MinA	54.46	98%	49%
		fixed axioms, ≤ 10 IAS/RAS	16.65	100%	96%
		fixed axioms, partial, ≤ 10 IAS/RAS	8.00	100%	99%
		fixed axioms, partial, lab-opt.	7.56	100%	100%
O^{SNOMED}	nonlinear	full PP, ≤ 10 MinA	184.76	100%	75%
		fixed axioms, ≤ 10 IAS/RAS	16.00	100%	99%
		fixed axioms, partial, ≤ 10 IAS/RAS	9.81	100%	100%
		fixed axioms, partial, lab-opt., ≤ 10 IAS/RAS	11.70	100%	100%
	linear	full PP, ≤ 10 MinAs	185.35	100%	75%
		fixed axioms, ≤ 10 IAS/RAS	41.66	100%	95%
		fixed axioms, partial, ≤ 10 IAS/RAS	27.50	100%	98%
		fixed axioms, partial, lab-opt., ≤ 10 IAS/RAS	32.80	100%	98%

extracted reachability-modules so that the search space is further reduced already. To conclude, fixed sub-ontologies and cardinality limit are optimizations with reasonable impact.

6 Conclusions

Previous work has studied labeled ontologies and methods to compute boundaries for their consequences. In the present paper we have looked at the problem of finding an adequate relabeling of the ontology in case that the boundary obtained differs from the desired one. Our approach focuses on the search of a change set of minimum cardinality. We identified simple cases, namely where the boundary ℓ_c and the goal label ℓ_g are comparable w.r.t. to the lattice ordering, in which known methods from ontology repair can be adapted. In particular, we showed that if $\ell_c < \ell_g$, then any MinA yields a change set, while if $\ell_g < \ell_c$, then diagnoses correspond to change sets. The remaining case, where ℓ_c and ℓ_g are incomparable is reduced to the previous one by using $\ell_c \otimes \ell_g$ as an intermediate goal label.

In order to find a change set with minimum cardinality, we presented a variation of the HST algorithm used in axiom-pinpointing. Our variations are based on three insights: (i) some axioms are irrelevant for the computation of a change set; thus, they can be removed from the search space from the beginning; (ii) since we are interested only in a change set of minimum cardinality, we can improve the search by avoiding unnecessarily large solutions; and (iii) axioms can

be grouped by their labels to first reduce the ontology from which the change set will be computed. We implemented algorithms to test the benefit of using the optimizations obtained by these three insights, and tested them on large-scale ontologies. Our first results show that the first two ideas yield tangible improvements in both the execution time and the quality of the solution. However, trying to optimize the search through the labels does not seem to pay off. All our algorithms are black-box based, which means that they can be used with any off-the-shelf reasoner, without the need of modifications.

As future work we intend to study the problem of finding change sets for several consequences (each with its own goal label) simultaneously. An availability policy could further restrict the set of axioms and their allowed label changes. We will also look more closely at the case where the boundary and the goal label are incomparable, and try to develop methods that improve both, the quality of the change set and its computation time. Moreover, we will look at other criteria for the minimality of change sets and more flexible restrictions on the goal label.

References

1. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope. In: Proc. of IJCAI'05, Edinburgh, UK (2005)
2. Baader, F., Knechtel, M., Peñaloza, R.: A generic approach for large-scale ontological reasoning in the presence of access restrictions to the ontology's axioms. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 49–64. Springer, Heidelberg (2009)
3. Baader, F., Peñaloza, R.: Automata-based axiom pinpointing. Journal of Automated Reasoning (to appear, 2010); Special Issue: IJCAR 2008 (2008)
4. Baader, F., Peñaloza, R.: Axiom pinpointing in general tableaux. Journal of Logic and Computation 20(1), 5–34 (2010); Special Issue: Tableaux'07
5. Baader, F., Peñaloza, R., Suntisrivaraporn, B.: Pinpointing in the Description Logic $\mathcal{EL}^+$. In: Hertzberg, J., Beetz, M., Englert, R. (eds.) KI 2007. LNCS (LNAI), vol. 4667, pp. 52–67. Springer, Heidelberg (2007)
6. Baader, F., Suntisrivaraporn, B.: Debugging SNOMED CT using axiom pinpointing in the description logic $\mathcal{EL}^+$. In: Proc. of KR-MED'08 (2008)
7. Biskup, J., Embley, D.W., Lochner, J.-H.: Reducing inference control to access control for normalized database schemas. Inf. Process. Lett. 106(1), 8–12 (2008)
8. Farkas, C., Jajodia, S.: The inference problem: a survey. SIGKDD Explor. Newsl. 4(2), 6–11 (2002)
9. Gaag, A., Kohn, A., Lindemann, U.: Function-based solution retrieval and semantic search in mechanical engineering. Proc. of ICED'09 (2009)
10. Kalyanpur, A., Parsia, B., Horridge, M., Sirin, E.: Finding all justifications of OWL DL entailments. In: Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L.J.B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) ASWC 2007 and ISWC 2007. LNCS, vol. 4825, pp. 267–280. Springer, Heidelberg (2007)

11. Kalyanpur, A., Parsia, B., Sirin, E., Hendler, J.A.: Debugging unsatisfiable classes in OWL ontologies. J. Web Sem. 3(4), 268–293 (2005)
12. Meyer, T., Lee, K., Booth, R., Pan, J.Z.: Finding maximally satisfiable terminologies for the description logic $\mathcal{ALC}$. In: Proc. of AAAI'06 (2006)
13. Schlobach, S., Cornet, R.: Non-standard reasoning services for the debugging of description logic terminologies. In: Proc. of IJCAI'03, pp. 355–362 (2003)
14. Suntisrivaraporn, B.: Polynomial-Time Reasoning Support for Design and Maintenance of Large-Scale Biomedical Ontologies. PhD thesis, TU Dresden (2008)