Correcting Access Restrictions to a Consequence More Flexibly

Eldora[1⋆], Martin Knechtel[2], and Rafael Peñaloza[1]

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

Abstract. Recent research has shown that labeling ontologies can be useful for restricting the access to some of the axioms and their implicit consequences. However, the labeling of the axioms is an error-prone and highly sensible task. In previous work we have shown how to correct the access restrictions if the security administrator knows the precise access level that a consequence must receive, and axioms are relabeled to that same access level. In this paper, we look at a more general situation in which access rights can be granted or denied to some specific users, without having to fully specify the precise access level. We also allow a more flexible labeling function, where the new access level of the relabeled axioms may differ from the level of the restriction. We provide black-box algorithms for computing suggestions of axioms to be relabeled.

1 Introduction

Description Logics (DL) [1] have been successfully used to represent knowledge of various application domains. One of the main advantages of using a logic-based knowledge representation language is the possibility of reasoning within the system; that is, deriving implicit consequences from the explicitly stated knowledge in the ontology.

In some application domains it is desirable to restrict users to access only portions of the ontology. For instance, in a security scenario [5], users with a low security clearance should not be able to access classified information. Other motivations for restricting access to users are the reduction of information overload, or filtering w.r.t. a level of specialization. Rather than maintaining different sub-ontologies for each definable user level, we have previously proposed [2] to label each axiom with information on which users can access it. Reasoning then generalizes to the task of finding an adequate label for each implicit consequence of the ontology. This label, called a boundary, can be computed through black-box [2] as well as glass-box [9] techniques.

However, the task of labeling axioms according to their access level is error-prone and highly sensitive to noise. Indeed, a set of seemingly innocuous axioms

⋆ This work was developed while the author worked for SAP Research Dresden.

may allow a user to derive some unwanted consequence. Dually, a too restrictive access level may hide a consequence from relevant users. This problem becomes more pronounced if neither the security administrator nor the knowledge engineer is an expert in logic. We thus want to develop a system that can automatically suggest changes in the labeling function that correct the access to a given consequence.

In previous work [8, 7] we have developed and implemented efficient algorithms for correcting access restrictions to implicit consequences if (i) the knowledge engineer knows the exact access level the consequence must receive (called the goal label) and (ii) axioms are always relabeled to the goal label. In this paper we relax these both conditions. On the one hand, we allow the knowledge engineer to specify a bound on the desired access level, rather than an exact value. This is useful, for instance, to express that a set of users must all have access to the consequence, but it is irrelevant which other users (if any) can also derive it. On the other hand, the knowledge engineer is also able to specify a so-called target label to which the axioms are relabeled. Contrary to the previous approach, the target label needs not be equal to the goal label.

We develop black-box algorithms for finding the minimal sets of axioms that need to be relabeled to the target label in order for the access of the consequence to satisfy the restriction imposed. Additionally, we show that our methods can be improved if one is only interested in finding one such set of minimal cardinality. All our methods are based on results and ideas from axiom-pinpointing [11, 3], but optimized by considering the labels of the axioms used.

2 Preliminaries

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*. An ontology language specifies which sets of axioms are admitted as ontologies, with the only restriction 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 $\mathcal{O}$. 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. Consider, for instance, a description logic $\mathcal{L}$. Then, an ontology is a finite set of general concept inclusion axioms (GCIs) of the form $C \sqsubseteq D$, with C, D $\mathcal{L}$-concept descriptions and assertion axioms of the form $C(b)$, with C an $\mathcal{L}$-concept description and b an individual name. An example of a consequence is the subsumption relation $A \sqsubseteq B$ between concept names A, B.

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 a user labeled with the access level $\ell \in L$, we denote as $\mathcal{O}_{\geq \ell}$ the sub-ontology $\mathcal{O}_{\geq \ell} := \{a \in \mathcal{O} \mid \mathsf{lab}(a) \geq \ell\}$ visible for him. 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. Conversely, for a sub-ontology $\mathcal{S} \subseteq \mathcal{O}$, we define

$$\lambda_{\mathcal{S}} := \bigotimes_{a \in \mathcal{S}} \mathsf{lab}(a) \text{ and } \mu_{\mathcal{S}} := \bigoplus_{a \in \mathcal{S}} \mathsf{lab}(a).$$

An element $\ell \in L$ is *join prime relative to* L_{lab} if for every $K_1, \ldots, K_n \subseteq L_{\mathsf{lab}}$, it holds that $\ell \leq \bigoplus_{i=1}^{n} \lambda_{K_i}$ implies that there is $i, 1 \leq i \leq n$ such that $\ell \leq \lambda_{K_i}$. For instance, in the lattice from Figure 1, ℓ_1 and ℓ_4 are the only elements that are not join prime relative to $L_{\mathsf{lab}} = \{\ell_1, \ldots, \ell_5\}$, since $\ell_1 \leq \ell_2 \oplus \ell_4$ but neither $\ell_1 \leq \ell_2$ nor $\ell_1 \leq \ell_4$ and similarly $\ell_4 \leq \ell_5 \oplus \ell_3$ but neither $\ell_4 \leq \ell_5$ nor $\ell_4 \leq \ell_3$. 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, lab a labeling function and c a consequence. An element $\nu \in L$ is called a* boundary *for $\mathcal{O}$,c,lab 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 2. *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 relating the boundary with the set of diagnoses, also holds.

Lemma 3. *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 4. 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

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

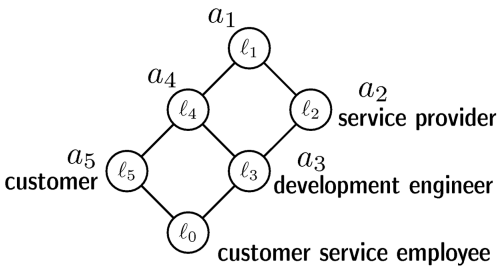

Fig. 1. Lattice $(L_d, \leq_d)$ with 4 user labels and an assignment of 5 axioms to labels

where the function lab assigns to each axiom a_i the label ℓ_i as shown in Figure 1. This ontology entails $\mathit{ServiceWithComingPriceIncrease}(\mathit{ecoCalculatorV1})$. The MinAs for $\mathcal{O}$,c are $\{a_1, a_2, a_3\}, \{a_1, a_2, a_5\}, \{a_1, a_3, a_4\}, \{a_1, a_4, a_5\}$, and its diagnoses are $\{a_1\}, \{a_2, a_4\}, \{a_3, a_5\}$. Using Lemma 3, we can compute the boundary as $\mu_{\{a_1\}} \otimes \mu_{\{a_2,a_4\}} \otimes \mu_{\{a_3,a_5\}} = \ell_1 \otimes \ell_1 \otimes \ell_4 = \ell_4$. The join prime elements relative to L_{lab}, which define valid user labels, are $\ell_0, \ell_2, \ell_3, \ell_5$. These labels represent the user roles as illustrated. Thus, the consequence c is only visible for the user roles ℓ_0, ℓ_5 and ℓ_3, i.e. for customer service employees, customers, and development engineers.

3 Modifying the Boundary

An efficient implementation of a black-box algorithm for computing the boundary of DL consequences already exists [2]. However, a desirable addition to this method is the capability of automatically relabeling some of the axioms to correct the access level of some implicit consequence. Indeed, labeling axioms w.r.t. their access restrictions is highly error-prone, and very small changes in the labeling function may produce consequences to become visible to unauthorized users, or inaccessible to the relevant users.

We have previously shown [8, 7] how to detect a set of axioms of minimal cardinality that needs to be relabeled for obtaining a given boundary. However, in that setting the knowledge engineer must specify the exact boundary that the consequence must receive, and all axioms are relabeled to that value. We now relax these restrictions, by allowing more general constraints on the new boundary, and a more flexible relabeling function.

Definition 5 (Boundary Constraint, Change Set). *A* boundary constraint *is a tuple $\beta = (c, \propto \ell_g, \ell_t)$, where c is a consequence, $\propto \ell_g$, with $\propto \in \{\leq, \geq, \not\geq, \not\leq\}$, $\ell_g \in L$ is a* condition *and ℓ_t is the target label with $\ell_t \propto \ell_g$.*

Let $\mathcal{O}$ be an ontology, $\mathcal{S} \subseteq \mathcal{O}$, lab a labeling function, and $\ell \in L$. We define the modified labeling function $\mathsf{lab}_{\mathcal{S},\ell}$ as

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

A sub-ontology $\mathcal{S} \subseteq \mathcal{O}$ is called a change set (CS) *for the boundary constraint $\beta = (c, \propto \ell_g, \ell_t)$ if the boundary ν for $\mathcal{O}, c, \mathsf{lab}_{\mathcal{S},\ell_t}$ satisfies $\nu \propto \ell_g$.*

For the rest of this paper, we assume, w.l.o.g. that the boundary for $\mathcal{O}, c, \mathsf{lab}$ does not satisfy the condition $\propto \ell_g$ since otherwise, the empty set is already a CS and nothing needs to be changed in the labeling function.

Notice that, since $\ell_t \propto \ell_g$, the whole ontology $\mathcal{O}$ is always a change set. However, using the whole ontology as a change set would set ℓ_t as the boundary of every consequence of $\mathcal{O}$. In general, we want to make the least possible changes when correcting the boundary of a given consequence. For that reason, we will focus on finding all those change sets that are minimal w.r.t. set inclusion. These sets are useful if the knowledge engineer wants to obtain several suggestions of correction, and then choose the adequate one by some external criterion. However, due to the huge number (possibly exponentially many [4]) of change sets that may exist, one may also look for the "best" change set, and use it automatically in the correction. Hence, we also study how to find a *smallest* change set; that is, one with the least cardinality.

We divide this section in two parts. First we look at the case where the boundary restriction is of the form $\leq$ or $\geq$. We show that previously known techniques can be used also in this setting. We then look at the negative restrictions, which require new methods to be developed.

3.1 Positive Conditions

We now focus on the case where the condition of the boundary constraint is of the form $\geq \ell_g$. Due to the duality of MinAs and diagnoses, the case for $\leq \ell_g$ can be treated in an analogous way (see e.g. [8]).

Let $\beta = (c, \geq \ell_g, \ell_t)$ be a boundary constraint and $\ell_t \geq \ell_g$. Recall (Lemma 2) that the boundary can be computed as the supremum of all $\lambda_{\mathcal{S}_i}$, where $\mathcal{S}_i$ is a MinA for $\mathcal{O}, c$. Thus, if we relabel all the axioms in a MinA $\mathcal{S}$ to ℓ_t, then the boundary for $\mathcal{O}, c, \mathsf{lab}_{\mathcal{S},\ell_t}$ is $\geq \ell_t \geq \ell_g$; that is, every MinA is a change set. Yet, this change set may not be minimal. In fact, we only need that the infimum of the labels of all the axioms in this MinA is $\geq \ell_g$. This can be achieved by only relabeling the axioms in $\mathcal{S}$ that are not already $\geq \ell_g$.

Example 6. Continuing Example 4, recall that we have computed the label ℓ_4 as the boundary of the consequence c. Suppose now that we want to change this boundary to be $\geq \ell_2$, using ℓ_2 also as the relabeling target. As described above, every MinA is also a change set for this consequence. If we consider the MinA $\mathcal{S} = \{a_1, a_2, a_3\}$, then under the new labeling $\mathsf{lab}_{\mathcal{S},\ell_2}$ we obtain the new boundary

$$\lambda_{\{a_1,a_2,a_3\}} \oplus \lambda_{\{a_1,a_2,a_5\}} \oplus \lambda_{\{a_1,a_3,a_4\}} \oplus \lambda_{\{a_1,a_4,a_5\}} = \ell_2 \oplus \ell_0 \oplus \ell_3 \oplus \ell_0 = \ell_2.$$

However, it is easy to see through a simple computation, that the set $\{a_3\}$ is also a change set, which is strictly included in the previous MinA. This set is obtained from the MinA by removing all axioms whose label is greater or equal ℓ_2, namely a_1 and a_2.

Intuitively, we simply consider every axiom $a \in \mathcal{O}$ with $\mathsf{lab}(a) \geq \ell_g$ as *fixed* in the sense that its label cannot be changed, as changing it will be superfluous for any CS. We thus consider a generalization of MinAs, called IAS.

Definition 7 (IAS). *A* minimal inserted axiom set (IAS) *for ℓ is a subset $I \subseteq \mathcal{O}$ such that $\mathcal{O}_{\geq \ell} \cup I \models c$ and $\mathcal{O}_{\geq \ell} \cup I' \not\models c$ for all $I' \subset I$.*

The known algorithms for computing all MinAs [6, 12] through a hitting set tree (HST) method [10] can easily be adapted for also computing IAS [8]. More interestingly, the set of all minimal change sets corresponds to the set of all IAS.

Theorem 8. *Let $\mathcal{O}$ be an ontology, $\beta = (c, \geq \ell_g, \ell_t)$ a boundary constraint and $\mathcal{S} \subseteq \mathcal{O}$. $\mathcal{S}$ is a minimal CS for β iff $\mathcal{S}$ is an IAS for ℓ_g.*

In [8, 7] it is shown how to compute the set of all IAS for a consequence c. Moreover, the algorithms presented there have been also optimized for finding the smallest IAS, through the inclusion of a cardinality restriction. Basically, the construction of an IAS stops once that this has reached the cardinality of the smallest IAS found so far. It was shown that using these (partial) IAS can drastically reduce the search space, while preserving correctness of the method. Due to Theorem 8, all the algorithms for computing IAS and IAS of minimal cardinality can be used for finding the minimal change sets and a change set of minimal cardinality, for positive boundary constraints.

3.2 Negative Conditions

We now consider the case in which the boundary constraint has a condition of the form $\not\geq \ell_g$. As in the previous section, the case for $\not\leq \ell_g$ can be solved dually by simply interchanging MinAs and diagnoses.

Given an ontology $\mathcal{O}$, a labeling function lab and a consequence c, if the boundary for $\mathcal{O}, c, \mathsf{lab}$ is greater or equal to ℓ_g, then we know that for every diagnosis $\mathcal{S}$ for $\mathcal{O}, c$ it holds that $\mu_{\mathcal{S}} \geq \ell_g$ (see Lemma 3). Hence, if we relabel all the axioms in any diagnosis $\mathcal{S}$ to $\ell_t \not\geq \ell_g$, it follows that the boundary is then changed to a new value $\not\geq \ell_g$; that is, $\mathcal{S}$ is a CS. However, just as in the previous section, this CS may not be minimal. One idea to try to find a minimal CS is to follow the same intuition as in the previous section, and fix all axioms whose labels already satisfy the condition $\not\geq \ell_g$. Unfortunately, this idea is not correct, as shown by the following example.

Example 9. Returning to Example 4, suppose now that we want to change the boundary from ℓ_4 to some value $\not\geq \ell_4$, using ℓ_5 as a target label. Recall that $\{a_2, a_4\}$ and $\{a_3, a_5\}$ are diagnoses for the consequence. If we consider the axioms having a label $\not\geq \ell_4$ as fixed, then none of these diagnoses produces a change set. In the first one, the axiom a_2 would be fixed, but then, under the relabeling $\mathsf{lab}_{\{a_4\},\ell_5}$ we will obtain the boundary

$$\mu_{\{a_1\}} \otimes \mu_{\{a_2,a_4\}} \otimes \mu_{\{a_3,a_5\}} = \ell_1 \otimes (\ell_2 \oplus \ell_5) \otimes (\ell_3 \oplus \ell_5) = \ell_1 \otimes \ell_1 \otimes \ell_4 = \ell_4,$$

Algorithm 1 Compute one minimal CS contained in a diagnosis

Procedure compute-one-CS($\mathcal{S}, \beta$)
Input: $\mathcal{S}$: diagnosis; $\beta = (c, \not\geq \ell_g, \ell_t)$: boundary constraint;
Output: $\mathcal{T} \subseteq \mathcal{S}$: minimal CS for β contained in $\mathcal{S}$

```
1: if ℓ_t ≥ ℓ_g then
2:    return ∅
3: T := S
4: ℓ := ℓ_t
5: for every a ∈ S do
6:    if ℓ ⊕ lab(a) ≱ ℓ_g then
7:       T := T \ {a}
8:       ℓ := ℓ ⊕ lab(a)
9: return T
```

which does not satisfy the restriction $\not\geq \ell_4$; hence, $\{a_4\}$ is not a change set.

In the case of the second diagnosis, the problem is even greater, since both axioms will be considered as fixed. Thus, the approach would deduce that no axiom needs to be relabeled to obtain a boundary $\not\geq \ell_4$, which is obviously not true.

Despite this, it is still possible to use diagnoses as a basis for computing the minimal CS. Suppose that we have a diagnosis $\mathcal{S}$ containing an axiom a_0 such that $\ell_t \oplus \mathsf{lab}(a_0) \not\geq \ell_g$. Then, $\mathcal{S}' = \mathcal{S} \setminus \{a_0\}$ is also a CS, since

$$\bigoplus_{a \in \mathcal{S}} \mathsf{lab}_{\mathcal{S}', \ell_t}(a) = \ell_t \oplus \mathsf{lab}(a_0) \not\geq \ell_g.$$

Obviously, this result holds not only for a single axiom a_0 but for any subset $\mathcal{T}$ of $\mathcal{S}$ such that $\ell_t \oplus \bigoplus_{a \in \mathcal{T}} \mathsf{lab}(a) \not\geq \ell_g$.

Lemma 10. *Let $\mathcal{S}$ be a diagnosis for $\mathcal{O}, c$ and $\beta = (c, \not\geq \ell_g, \ell_t)$ a boundary constraint. If $\mathcal{T}$ is a subset of $\mathcal{S}$ such that $\ell_t \oplus \bigoplus_{a \in \mathcal{T}} \mathsf{lab}(a) \not\geq \ell_g$, then $\mathcal{S} \setminus \mathcal{T}$ is a CS for β.*

Proof. For every axiom $a \in \mathcal{S} \setminus \mathcal{T}$, $\mathsf{lab}_{\mathcal{S} \setminus \mathcal{T}, \ell_t}(a) = \ell_t$. Additionally, we know that $\bigoplus_{a \in \mathcal{S}} \mathsf{lab}(a) \geq \ell_g$, and hence $\mathcal{T} \neq \mathcal{S}$. Thus, under the new labeling, we have that

$$\bigoplus_{a \in \mathcal{S}} \mathsf{lab}_{\mathcal{S} \setminus \mathcal{T}, \ell_t}(a) = \ell_t \oplus \bigoplus_{a \in \mathcal{T}} \mathsf{lab}(a) \not\geq \ell_g$$

Since $\mathcal{S}$ is a diagnosis, Lemma 3 implies that the new boundary satisfies the condition, and hence $\mathcal{S} \setminus \mathcal{T}$ is a CS. □

A simple consequence of this lemma is that, given a maximal subset $\mathcal{T}$ of $\mathcal{S}$ satisfying $\ell_t \oplus \bigoplus_{a \in \mathcal{T}} \mathsf{lab}(a) \not\geq \ell_g$, $\mathcal{S} \setminus \mathcal{T}$ is a minimal change set for β *contained* in $\mathcal{S}$. Algorithm 1 describes how to compute one such minimal change set from a diagnosis. This, however, might not be a "globally" minimal change set; that is, there might still exist other change sets strictly contained in it, as shown in the following example.

Example 11. Consider the lattice in Figure 1, an ontology $\mathcal{O}$ having four axioms $\{a_1, a_2, a_3, a_4\}$, and a consequence c such that the diagnoses for $\mathcal{O}, c$ are the sets $\{a_1, a_2, a_3\}$ and $\{a_1, a_4\}$. Assume that the labeling function lab is given by the mapping $\mathsf{lab}(a_1) = \ell_4, \mathsf{lab}(a_2) = \ell_5, \mathsf{lab}(a_3) = \mathsf{lab}(a_4) = \ell_2$. It is easy to see that the boundary for this consequence is ℓ_1. If we apply Algorithm 1 to the diagnosis $\{a_1, a_2, a_3\}$ and the boundary constraint $\beta = (c, \not\geq \ell_1, \ell_3)$, where at Line 5, we first choose a_3, then ℓ is changed to ℓ_2 at Line 8, and hence the test $\ell \oplus \mathsf{lab}(a) \not\geq \ell_1$ fails for axioms a_1 and a_2. Thus, the algorithm returns the change set $\{a_1, a_2\}$. However, $\{a_1\}$ is also a change set, since if a_1 is relabeled to ℓ_3, then $\mu_{\{a_1,a_4\}} = \ell_2$,and thus the boundary is $\not\geq \ell_1$.

Although Algorithm 1 does not always output a globally minimal change set, one can still use it for computing all the minimal change sets for β. The idea is based on the following lemma, which is a simple consequence of the definition of diagnoses and change sets.

Lemma 12. *Let $\mathcal{S}$ be a minimal change set for $(c, \not\geq \ell_g, \ell_t)$. Then, there exists a set $\mathcal{T}$ such that (i) $\ell_t \oplus \bigoplus_{a\in\mathcal{T}} \mathsf{lab}(a) \not\geq \ell_g$ and (ii) $\mathcal{S} \cup \mathcal{T}$ is a diagnosis for $\mathcal{O}, c$.*

For instance, in Example 11 we found the minimal change set $\{a_1\}$. The set $\mathcal{T} = \{a_4\}$ satisfies the two conditions stated in Lemma 12.

To compute all minimal change sets, one then needs to compute all diagnoses, and from each of these diagnoses compute all the minimal change sets that are contained in it. This is possible through a nesting of two hitting set tree (HST) algorithms: the external one produces all different diagnoses for $\mathcal{O}, c$, while the internal generates, for any given diagnosis, all the maximal subsets of axioms that can be removed to obtain a CS. Algorithm 2 shows how this internal HST algorithm works.

The idea behind all HST-like algorithms is the following. One first computes a set of axioms $\mathcal{T}$ satisfying some property; in the case of Algorithm 2, the set is a minimal CS for β contained in $\mathcal{S}$. This set is then used to label the root of the tree. The algorithm then branches as follows. For each axiom a in $\mathcal{T}$, a new branch is created and a is removed from the search space. A new set $\mathcal{T}'$ satisfying the property is then computed, and used to label the successor node. The removal of the axiom $a \in \mathcal{T}$ from the search space ensures that $\mathcal{T} \not\subseteq \mathcal{T}'$. This process is then iterated until the property is not satisfied by the search space; that is, Algorithm 1 returns the empty set. This process stops after at most exponentially many iterations, on the size of $\mathcal{S}$, and the labels of the tree contain all the minimal sets of axioms satisfying the property; in our case, all minimal change sets contained in the diagnosis.

There are two common optimizations for HST algorithms, which are also used in Algorithm 2. The first one is called *early path termination*. The idea behind this optimization is that if one can distinguish parts of the tree that will yield no new minimal sets of axioms, then one can stop exploring those branches. The two conditions for early path termination described in Line 1 of expand-hst test for a path where the search space is contained in a search space already explored in a

Algorithm 2 HST to compute all minimal CS contained in a diagnosis

Procedure compute-all-CS$(\mathcal{S}, \beta)$
Input: $\mathcal{S}$: diagnosis; $\beta = (c, \not\gtrsim \ell_g, \ell_t)$: boundary constraint;
Output: **C**: all minimal CS for β contained in $\mathcal{S}$

1: **Global C, H** $:= \emptyset$
2: $\mathcal{T}$:=compute-one-CS$(\mathcal{S}, \beta)$
3: **C** $:= \{\mathcal{T}\}$
4: **for** each $a \in \mathcal{T}$ **do**
5: expand-hst$(\mathcal{S}, (c, \not\gtrsim \ell_g, \ell_t \oplus \mathsf{lab}(a)), \{a\})$
6: **return C**

Procedure expand-hst$(\mathcal{S}, \beta, H)$
Input: $\mathcal{S}$: diagnosis; $\beta = (c, \not\gtrsim \ell_g, \ell_t)$: boundary constraint; H: list of axioms
Side effects: modifications to **C**,**H**

1: **if** exists some $H' \in \mathbf{H}$ such that $H' \subseteq H$ **or**
H' contains a prefix path P with $P = H$ **then**
2: **return** (early path termination)
3: $\mathcal{T}' := \emptyset$
4: **if** exists some $\mathcal{T} \in \mathbf{C}$ such that $\ell_t \oplus \bigoplus_{a \in \mathcal{S} \setminus \mathcal{T}} \mathsf{lab}(a) \propto \ell_g$ **then**
5: $\mathcal{T}' := \mathcal{T}$ (CS reuse)
6: **else**
7: $\mathcal{T}'$:=compute-one-CS$(\mathcal{S}, \beta)$
8: **if** $\mathcal{T}' \neq \emptyset$ **then**
9: **C** $:=$ **C** $\cup \{\mathcal{T}'\}$
10: **for** each $a \in \mathcal{T}'$ **do**
11: expand-hst$(\mathcal{S}, (c, \not\gtrsim \ell_g, \ell_t \oplus \mathsf{lab}(a)), H \cup \{a\})$
12: **else**
13: **H** $:=$ **H** $\cup \{H\}$ (normal termination)

previous branch. The second optimization is the reuse of sets. When expanding a tree, we only ask for a set of axioms satisfying the property that is contained in the current search space. If these conditions hold in a previously computed label, then we can reuse it, avoiding this way a possibly expensive call to Algorithm 1.

To find all "global" minimal change sets, we use an additional HST algorithm that computes all diagnoses, and for each of these, calls Algorithm 2. This algorithm uses the same kind of optimizations. However, to improve the functionality of the reuse of solutions, the set of all change sets computed so far is kept in a global variable, accessible from every call to compute-all-CS. Thus, a change set that has been previously computed from a diagnosis $\mathcal{S}$, can be reused in a call with a different diagnosis $\mathcal{S}'$.

It is worth noticing that in some cases, a diagnosis may contain several axioms labeled with the same lattice element. Moreover, the condition for obtaining a minimal CS from Lemma 10 depends only on the labeling, and not in the axiom itself. Thus, it is sometimes possible to optimize the search for the minimal CS by considering only the labels and not the individual axioms, as described in Algorithm 3. The correctness of this algorithm is justified by the following lemma, whose proof is analogous to the one of Lemma 10.

Algorithm 3 Compute one minimal CS contained in a diagnosis (optimized)

Procedure compute-one-CS$(\mathcal{S}, \beta)$
Input: $\mathcal{S}$: diagnosis; $\beta = (c, \not\geq \ell_g, \ell_t)$: boundary constraint;
Output: $\mathcal{T} \subseteq \mathcal{S}$: minimal CS for β

1: **if** $\ell_t \geq \ell_g$ **then**
2: **return** no CS
3: $\mathcal{T} := \mathcal{S}$
4: $\ell := \ell_t$
5: $L := \{\mathsf{lab}(a) \mid a \in \mathcal{S}\}$
6: **for** every $m \in L$ **do**
7: **if** $\ell \oplus m \not\geq \ell_g$ **then**
8: $\mathcal{T} := \mathcal{T} \setminus \{a \mid \mathsf{lab}(a) = m\}$
9: $\ell := \ell \oplus m$
10: **return** $\mathcal{T}$

Lemma 13. *Let $\mathcal{S}$ be a diagnosis for $\mathcal{O}, c$, $\beta = (c, \not\geq \ell_g, \ell_t)$ a boundary constraint, and $L_{\mathcal{S}} = \{\mathsf{lab}(a) \mid a \in \mathcal{S}\}$. If $\mathcal{M} \subseteq L_{\mathcal{S}}$ is such that $\ell_t \oplus \bigoplus_{\ell \in \mathcal{M}} \ell \not\geq \ell_g$, then $\mathcal{S} \setminus \{a \mid \mathsf{lab}(a) \in \mathcal{M}\}$ is a CS for β.*

As in the case for positive conditions, these algorithms can be further optimized if one is only interested in a change set of minimal cardinality. Notice simply that in Algorithms 1 and 3, whenever the condition in the **for** loop is violated, then at least an axiom is ensured to belong to the output change set. Thus, it is easy to adapt these algorithms to include a cardinality bound, returning a partial CS once it has reached a given size. Since our method uses an HST approach, the proofs of correctness of the variant of HST capable of exploiting cardinality restrictions [8] hold also in this case. In other words, Algorithm 2 can be further optimized to compute only one change set of minimal cardinality.

4 Conclusions

We have presented algorithms for correcting the boundary of a consequence in a more flexible manner than previous approaches. Our framework allows the knowledge engineer to set bounds on what the new boundary should be, and specify a label as the target of the relabeling. This flexibility is useful if, for instance, she wants to grant access to a consequence to some user, but is not willing to specify the exact set of users that should access it.

We developed algorithms that output all the minimal change sets. Additionally, we show how these algorithms can be optimized if one is only interested in an arbitrary change set of minimal cardinality.

As future work, we will first implement and test the performance of our methods on large-scale real-world ontologies and applications. We also plan to generalize our framework to allow axioms to be relabeled to different elements of the lattice, according to an adequate minimality criterion.

References

1. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.
2. F. Baader, M. Knechtel, and R. Peñaloza. A generic approach for large-scale ontological reasoning in the presence of access restrictions to the ontology's axioms. In A. B. et al., editor, *Proceedings of the 8th International Semantic Web Conference (ISWC 2009)*, Washington, DC, 2009.
3. F. Baader and R. Peñaloza. Axiom pinpointing in general tableaux. *Journal of Logic and Computation*, 20(1):5–34, February 2010. Special Issue: Tableaux and Analytic Proof Methods.
4. F. Baader, R. Peñaloza, and B. Suntisrivaraporn. Pinpointing in the description logic $\mathcal{EL}^+$. In J. Hertzberg, M. Beetz, and R. Englert, editors, *Proceedings of the 30th German Annual Conference on Artificial Intelligence (KI'07)*, volume 4667 of *Lecture Notes in Artificial Intelligence*, pages 52–67, Osnabrück, Germany, 2007. Springer-Verlag.
5. C. Farkas and S. Jajodia. The inference problem: a survey. *SIGKDD Explor. Newsl.*, 4(2):6–11, 2002.
6. A. Kalyanpur, B. Parsia, M. Horridge, and E. Sirin. Finding all justifications of OWL DL entailments. In K. Aberer, K.-S. Choi, N. F. Noy, D. Allemang, K.-I. Lee, L. J. B. Nixon, J. Golbeck, P. Mika, D. Maynard, R. Mizoguchi, G. Schreiber, and P. Cudré-Mauroux, editors, *Proc. of the 6th Int. Semantic Web Conf. and 2nd Asian Semantic Web Conf. (ISWC'07,ASWC'07)*, volume 4825 of *LNCS*, pages 267–280, Busan, Korea, 2007. Springer-Verlag.
7. M. Knechtel and R. Peñaloza. Correcting access restrictions to a consequence. In V. Haarslev, D. Toman, and G. Weddell, editors, *Proceedings of the 2010 International Workshop on Description Logics (DL'10)*, volume 573 of *CEUR-WS*, Waterloo, Canada, 2010.
8. M. Knechtel and R. Peñaloza. A generic approach for correcting access restrictions to a consequence. In L. Aroyo, G. Antoniou, E. Hyvönen, A. ten Teije, H. Stuckenschmidt, L. Cabral, and T. Tudorache, editors, *Proceedings of the 7th Extended Semantic Web Conference (ESWC 2010)*, volume 6088 of *Lecture Notes in Computer Science*, pages 167–182, 2010.
9. R. Peñaloza. Using sums-of-products for non-standard reasoning. In A.-H. Dediu, H. Fernau, and C. Martín-Vide, editors, *Proceedings of the 4th International Conference on Language and Automata Theory and Applications (LATA 2010)*, volume 6031 of *Lecture Notes in Computer Science*, pages 488–499. Springer-Verlag, 2010.
10. R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95, 1987.
11. S. Schlobach and R. Cornet. Non-standard reasoning services for the debugging of description logic terminologies. In G. Gottlob and T. Walsh, editors, *Proc. of the 18th Int. Joint Conf. on Artificial Intelligence (IJCAI'03)*, pages 355–362, Acapulco, Mexico, 2003. Morgan Kaufmann, Los Altos.
12. B. Suntisrivaraporn. *Polynomial-time Reasoning Support for Design and Maintenance of Large-scale Biomedical Ontologies*. PhD thesis, Technische Universität Dresden, 2009.