# Iterative Ontology Updates Using Context Labels

**Rafael Peñaloza**
KRDB Research Centre
Free University of Bozen-Bolzano, Italy
rafael.penaloza@unibz.it

**Aparna Saisree Thuluva**
Theoretical Computer Science
Technische Universität Dresden, Germany
aparna.thuluva@gmail.com

## Abstract

In the process of ontology management, it is important to be able to add or remove consequences, while preserving as much of the original ontology as possible. When these updates are made iteratively, this minimum change criterion is harder to satisfy. We propose a context-based method that stores the information about all the possible outcomes of an update compactly. Using it, we are guaranteed to find all the optimal solutions for an iterative update problem, and reason directly over them. We present a prototype implementation that can be used as a Protégé plug-in.

## 1 Introduction

Although ontology update operators and their properties are relatively well-understood, their study is usually limited to a single update operation in which a potentially implicit piece of knowledge is either removed or added to the ontology. However, the restriction to a single update is not realistic in many real-life applications, in which the knowledge engineers need to produce new versions of an ontology that reflect newly acquired knowledge. In this case, many updates are necessary; moreover, the outcome of earlier updates may influence the choice of the successive steps to follow. Thus, we can neither predict the future chain of updates, nor summarize the history of steps into a single update step. For this reason, it becomes important to study the properties of iterative applications of update operators.

When dealing with updates of a knowledge base or ontology, people are usually interested in satisfying the AGM postulates [Alchourrón, Gärdenfors, and Makinson, 1985]. In a nutshell, these postulates describe a set of conditions that can be easily agreed that every meaningful update operator must satisfy. One particular notion behind these postulates, and most work in the literature is the principle of minimal change. This principle requires that every update performed in an ontology should change the explicit knowledge as little as possible to obtain the desired result. Unfortunately, it is well-known that an iterative application of update operations might not preserve the postulates [Darwiche and Pearl, 1997]; in particular, a bigger change in the ontology at an early stage might lead to a smaller change overall.

In this paper, we propose a method for guaranteeing that the principle of minimal change is preserved through an iterative update process. The main idea behind our approach is in fact very simple: we simply store all the possible solutions, independently of whether they are among the best found so far or not. Since we also preserve sub-optimal solutions from the early stages of the update chain, if any of these yield better solutions on the long run, we are able to identify this situation, and output the adequate answer.

Obviously, a naïve application of this idea is unlikely to work in practice, as it requires to store and reason over potentially exponentially many different ontologies. For that reason, we propose to use labeled ontologies, which can be used to represent large classes of ontologies, called contexts, in a compact way [Ludwig and Peñaloza, 2014; Ceylan and Peñaloza, 2014]. We develop effective algorithms that manipulate these labeled ontologies to simulate the application of update operators on the contexts independently. These algorithms are based on the computation of the so-called boundary, which expresses the class of contexts that entail a given consequence.

All our algorithms were implemented in a system capable of performing iterative updates in an ontology written in extensions of the description logic $\mathcal{EL}$ [Baader, 2003]. Although limited in expressivity, this language is important as it has been used for developing many large ontologies, and is the logical basis of the OWL 2 EL profile of the standard ontology language for the semantic web.[1] We analyse the properties of our approach, and identify elements where future optimizations may lead to a better performance.

## 2 Ontology Languages

To keep our approach as general as possible, we consider an arbitrary ontology language as defined in [Baader, Knechtel, and Peñaloza, 2012]. In a nutshell, given two countable sets $\mathfrak{A}$ and $\mathfrak{C}$ of well-formed *axioms* and *consequences*, respectively, an *ontology language* defines (i) a class $\mathfrak{O}$ of finite subsets of $\mathfrak{A}$ such that for all $\mathcal{O}, \mathcal{O}' \subseteq \mathfrak{A}$, if $\mathcal{O} \in \mathfrak{O}$ and $\mathcal{O}' \subseteq \mathcal{O}$, then $\mathcal{O}' \in \mathfrak{O}$; and (ii) a binary relation $\models \subseteq \mathfrak{O} \times \mathfrak{C}$ such that for all $\mathcal{O}, \mathcal{O}' \in \mathfrak{O}$ and $c \in \mathfrak{C}$, if $\mathcal{O} \models c$ and $\mathcal{O} \subseteq \mathcal{O}'$, then $\mathcal{O}' \models c$. The elements of $\mathfrak{O}$ are called *ontologies*, and $\models$ is the *consequence relation*. If $\mathcal{O} \models c$, we say that $\mathcal{O}$ *entails* $c$.

[1] http://www.w3.org/TR/owl2-profiles/

Notice that, by definition, every subset of an ontology is also an ontology, and the consequence relation is monotonic.

Every Description Logic (DL) [Baader et al., 2007] is an ontology language in this sense. As an example, we consider the light-weight DL $\mathcal{EL}$ [Baader, Brandt, and Lutz, 2005]. $\mathcal{EL}$ *concepts* are build from the sets $\mathsf{N_C}$ and $\mathsf{N_R}$ of *concept names* and *role names*, respectively, using the grammar rule $C ::= A \mid \top \mid C \sqcap C \mid \exists r.C$, where $A \in \mathsf{N_C}$ and $r \in \mathsf{N_R}$. A *general concept inclusion* (GCI) is an expression $C \sqsubseteq D$, where $C, D$ are two concepts. In the case of $\mathcal{EL}$, the sets of axioms $\mathfrak{A}$ and of consequences $\mathfrak{C}$ coincide, and contain all possible GCIs; moreover, every finite subset of $\mathfrak{A}$ is an ontology. The consequence relation is defined with the help of interpretations. An *interpretation* is a pair $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ where $\Delta^{\mathcal{I}}$ is a non-empty set called the *domain*, and $\cdot^{\mathcal{I}}$ is an *interpretation function* mapping every $A \in \mathsf{N_C}$ to a subset $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and every $r \in \mathsf{N_R}$ to a binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The interpretation function is extended to concepts inductively by $\top^{\mathcal{I}} := \Delta^{\mathcal{I}}$, $(C \sqcap D)^{\mathcal{I}} := C^{\mathcal{I}} \cap D^{\mathcal{I}}$, and $(\exists r.C)^{\mathcal{I}} := \{d \in \Delta^{\mathcal{I}} \mid \exists e.(d, e) \in r^{\mathcal{I}}, e \in C^{\mathcal{I}}\}$. The interpretation $\mathcal{I}$ *satisfies* the GCI $C \sqsubseteq D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$; it is a *model* of the ontology $\mathcal{O}$ iff it satisfies all the GCIs in $\mathcal{O}$. The GCI $C \sqsubseteq D$ is a *consequence* of the ontology $\mathcal{O}$ ($\mathcal{O} \models C \sqsubseteq D$) iff every model of $\mathcal{O}$ also satisfies $C \sqsubseteq D$.

A *labeled ontology* $\mathcal{O}$ is simply an ontology in which every axiom $\alpha \in \mathcal{O}$ is associated to a set of labels $\mathsf{lab}(\alpha)$. Intuitively, a labeled ontology is a compact representation of a set of ontologies. More formally, let $\mathsf{lab}(\mathcal{O}) := \bigcup_{\alpha \in \mathcal{O}} \mathsf{lab}(\alpha)$; then every label $\ell \in \mathsf{lab}(\mathcal{O})$ defines the subontology $\mathcal{O}^{\mathsf{lab}}_{\ell}$ of axioms labelled with a set containing $\ell$; more precisely, $\mathcal{O}^{\mathsf{lab}}_{\ell} := \{\alpha \in \mathcal{O} \mid \ell \in \mathsf{lab}(\alpha)\}$. We call these subontologies *contexts*. The most relevant reasoning problem in the presence of labeled ontologies is to find out which contexts entail a given consequence. These are expressed through a boundary.

**Definition 1** (boundary)**.** Let $\mathcal{O}$ be a labeled ontology and $c$ a consequence. The *boundary* of $\mathcal{O}$ w.r.t. $c$ is the (unique) set $\nu \subseteq \mathsf{lab}(\mathcal{O})$ of labels such that $\mathcal{O}^{\mathsf{lab}}_{\ell} \models c$ iff $\ell \in \nu$.

In other words, computing the boundary allows us to reason over all the subontologies defined by the labeled ontology $\mathcal{O}$ simultaneously. Methods for computing this boundary based on Reiter's hitting set tree algorithm [Reiter, 1987] and the use of unmodified ontology reasoners have been developed and implemented.

## 3 Iterative Ontology Update

One important problem for the development and maintenance of ontologies is how to update them when new knowledge is acquired. Typically, updates can be of two kinds. One can either want to remove some knowledge (e.g., if it has been found to be incorrect), or to include some newly discovered knowledge. The former problem is usually called *contraction*, and the latter *expansion*. In an ontology language, given a consequence $c$ and an ontology $\mathcal{O}$, these problems refer to the task of modifying $\mathcal{O}$ into a new ontology $\mathcal{O}'$ such that $\mathcal{O}' \not\models c$ in the case of contraction, and $\mathcal{O}' \models c$ in the case of expansion. A variant of expansion, called *revision*, additionally requires that the new ontology $\mathcal{O}'$ remains consistent.

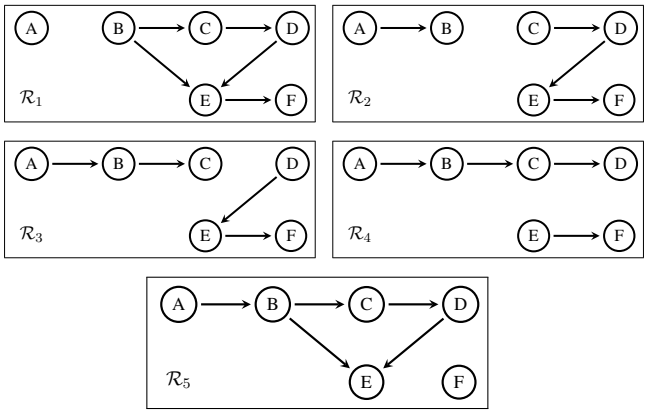


Figure 1: The repairs of $\mathcal{O}_{\mathsf{ex}}$ w.r.t. $A \sqsubseteq F$.

Clearly, one can conceive many different ways of achieving these results. To avoid arbitrary changes in an ontology, different desiderata on the properties of update operators have been proposed in terms of the AGM postulates [Alchourrón, Gärdenfors, and Makinson, 1985] and their successive variants.

One natural requirement is that the operators should follow the principle of minimal change; i.e., that the updated ontology should be as close to the original as possible, according to some similarity criterion. This is a fundamental criterion, which most update operators strive to fulfil. For example, to contract a consequence $c$ from the ontology $\mathcal{O}$, we can find a subontology $\mathcal{O}' \subseteq \mathcal{O}$ such that $\mathcal{O}' \not\models c$. To fulfil the principle of minimal change, we require this subontology to be maximal.

**Definition 2** (repair)**.** A *repair* of the ontology $\mathcal{O}$ w.r.t. the consequence $c$ is a subset $\mathcal{R} \subseteq \mathcal{O}$ such that $\mathcal{R} \not\models c$ and for all $\mathcal{R} \subset \mathcal{O}' \subseteq \mathcal{O}$, $\mathcal{O}' \models c$. We denote as $\mathsf{Rep}(\mathcal{O}, c)$ the set of all repairs of $\mathcal{O}$ w.r.t. $c$.

It is well known that there are potentially exponentially many repairs w.r.t. a single consequence. Thus, we can define a contraction operator that computes any repair of maximal size; that is, having the largest number of axioms.

**Example 3.** Suppose that we want to contract the consequence $A \sqsubseteq F$ from the $\mathcal{EL}$ ontology

$$\mathcal{O}_{\mathsf{ex}} := \{A \sqsubseteq B, B \sqsubseteq C, C \sqsubseteq D, D \sqsubseteq E, E \sqsubseteq F, B \sqsubseteq E\}.$$

The five repairs of $\mathcal{O}_{\mathsf{ex}}$ w.r.t. this consequence are depicted in Figure 1, where each arrow represents a GCI from $\mathcal{O}_{\mathsf{ex}}$. Three of these repairs ($\mathcal{R}_2$, $\mathcal{R}_3$, and $\mathcal{R}_4$) contain four axioms, while $\mathcal{R}_1$ and $\mathcal{R}_5$ contain five. Thus, to follow the principle of minimum change, any of the two repairs with maximum size may be chosen as a solution to the contraction problem.

To handle expansion, we will assume that only consequences that can also be expressed as axioms (i.e., only elements of $\mathfrak{A} \cap \mathfrak{C}$) can be expanded. Thus, to expand a consequence $c$ to $\mathcal{O}$, we need only to add $c$ to this ontology if $\mathcal{O}$ did not entail $c$ already.

In the process of updating an ontology, one should expect to perform several update operations, potentially depending

on the outcome of the previous results. Moreover, the consequences that will be retracted or expanded might not be known *a priori*. Thus, it is necessary to define iterative update operators. Unfortunately, a repeated application of an update operator is not guaranteed to satisfy the properties of a single update.

**Example 4.** Suppose that, after contracting $A \sqsubseteq F$ in Example 3, we wanted to contract $B \sqsubseteq E$. If in the first step we had chosen any of the maximal repairs ($\mathcal{R}_1$ or $\mathcal{R}_5$), then any answer to the second contraction step would have three axioms. Notice however that none of the other repairs, containing four axioms, entails $B \sqsubseteq E$, and hence are possible solutions to the iterative update problem. In fact, it is easy to see any repair of $\mathcal{R}_1$ or $\mathcal{R}_5$ w.r.t. $B \sqsubseteq E$ is a subset of one of either $\mathcal{R}_2$, $\mathcal{R}_3$, or $\mathcal{R}_4$.

The problem with iterative update is that there is no knowledge of what new update requests will come later. Thus, there is no way to guarantee that the chosen solution at an earlier step will lead to the best outcome in later stages. Moreover, since we do not know when the update process is over, we need to be ready to output the best solution at every step. For instance, in Example 4 we should be able to provide $\mathcal{R}_1$ or $\mathcal{R}_5$ as answer to the first contraction step, but $\mathcal{R}_2$, $\mathcal{R}_3$, or $\mathcal{R}_4$ after the second contraction.

An obvious way to achieve this goal is to preserve all the possible solutions over time. Thus, contracting $\mathcal{O}_{\mathsf{ex}}$ w.r.t. $A \sqsubseteq F$ yields the five ontologies $\mathcal{R}_1, \ldots, \mathcal{R}_5$; the second contraction step yields $\mathcal{R}_2, \ldots, \mathcal{R}_4$, since these contain the repairs of $\mathcal{R}_1$ and $\mathcal{R}_5$ w.r.t. $B \sqsubseteq E$, and so on. When a solution is required, one needs only to extract the best of the ontologies produced through this iteration.

The drawback of this approach is that it must store and update all the solutions at every step. Recall that an ontology may have exponentially many repairs w.r.t. a consequence. If we apply this idea, then we would have to preserve these exponentially many repairs. Moreover, at the next update step, the update operator would need to be applied to all of them. Obviously, this would result in very inefficient updates.

As described in the previous section, labeled ontologies represent a good choice for representing a set of ontologies compactly, and reasoning with them efficiently. Thus, we propose to improve on the approach sketched above by storing all the solutions found so far in a labeled ontology. More precisely, each solution will form a context in this ontology. We begin the update process with an unlabeled ontology, which we will see as a labeled ontology in which all axioms share the same label. When we retract a consequence $c$, we need to relabel the axioms in such a way that the new contexts are a non-redundant representation of the repairs of each of the previous contexts w.r.t. $c$. Dually, expansion means adding $c$ only to those contexts that did not entail $c$ already. In this view, we can consider revision as a special kind of iterative update in which we perform an expansion, followed by the contraction of inconsistencies. However, we need to be careful to guarantee that the revised axiom is not removed at the contraction step.

**Example 5.** Consider again the ontology $\mathcal{O}_{\mathsf{ex}}$ from Example 3, which is depicted as a labeled ontology in Figure 2 (a).

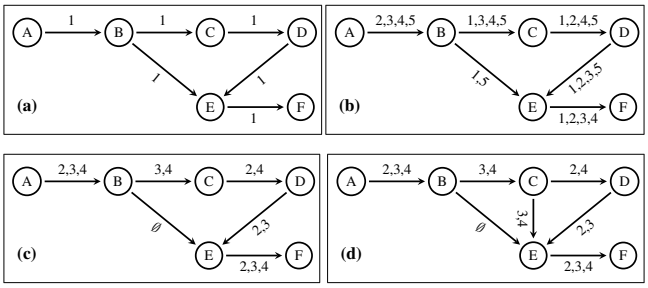


Figure 2: Labeled ontologies encoding all the solutions obtained from $\mathcal{O}_{\mathsf{ex}}$ (a) by iteratively retracting $A \sqsubseteq F$ (b), retracting $B \sqsubseteq E$ (c), and expanding $C \sqsubseteq E$ (d).

As we have seen, contracting the consequence $A \sqsubseteq F$ yields the five repairs from Figure 1, which can be described through the labeled ontology in Figure 2 (b). Further contracting $B \sqsubseteq E$ from these repairs yields the repairs $\mathcal{R}_2, \mathcal{R}_3, \mathcal{R}_4$ as depicted in part (c). To expand the consequence $C \sqsubseteq E$ it suffices to add this axiom to the labeled ontology with a label $3, 4$ expressing that only the contexts defined by these labels include that new axiom. Notice that the context $(\mathcal{O}_{\mathsf{ex}})^{\mathsf{lab}}_2$ already entails the consequence $C \sqsubseteq E$, and hence does not need to be expanded.

Obviously, this approach only makes sense if the new labels can be computed effectively. In the next sections, we describe methods for updating labeled ontologies, which we implemented in a tool that we describe later.

## 4 Labeled Contraction

For the rest of this paper, we assume that $\mathcal{O}$ is a labeled ontology whose contexts represent all the solutions to an iterative ontology update procedure at the current step. As mentioned before, this ontology might contain only one context, if no contraction has been made. We now show how to relabel this ontology to describe the result of contracting a given consequence.

Notice that in general we might have no information on the steps executed to reach the current labeling. Thus, the relabeling operator must depend exclusively on the current contexts, and not on the operations performed before hand. On the other hand, recall that the contraction operator that we defined in the previous section is based on the computation of repairs, which are by definition maximal w.r.t. set inclusion. To guide our methods, we will assume that the labeled ontology $\mathcal{O}$ is *irredundant*; that is, that for all labels $\ell, \ell' \in \mathsf{lab}(\mathcal{O})$, if $\ell \neq \ell'$, then $\mathcal{O}^{\mathsf{lab}}_{\ell} \not\subseteq \mathcal{O}^{\mathsf{lab}}_{\ell'}$.

Intuitively, the restriction to irredundant ontologies guarantees that no context represents a solution that is exactly the same as, or worse to, another context. Deciding whether an ontology $\mathcal{O}$ is irredundant needs only linear time in the size of $\mathcal{O}$. Moreover, a redundant ontology can be made irredundant by removing all contexts that are contained in any other context.

Given an ontology $\mathcal{O}$ whose axioms are labeled through the function $\mathsf{lab}$, and a consequence $c$, we are interested in

computing a new labeling function $\mathsf{lab}'$ that describes all the repairs of the contexts of the original ontology w.r.t. $c$. This is formalized next.

**Definition 6** (labeled contraction)**.** Let $\mathcal{O}$ be a labeled ontology with labeling function $\mathsf{lab}$, and $c \in \mathfrak{C}$. Define $\mathsf{Rep}(\mathcal{O}, \mathsf{lab}, c) := \bigcup_{\ell \in \mathsf{lab}(\mathcal{O})} \mathsf{Rep}(\mathcal{O}_\ell^{\mathsf{lab}}, c)$. We say that an element $\mathcal{R} \in \mathsf{Rep}(\mathcal{O}, \mathsf{lab}, c)$ is *maximal* if there is no other $\mathcal{R}' \in \mathsf{Rep}(\mathcal{O}, \mathsf{lab}, c)$ such that $\mathcal{R} \subset \mathcal{R}'$. A *labeled contraction* of $(\mathcal{O}, \mathsf{lab})$ w.r.t. $c$ is a labeling function $\mathsf{lab}'$ such that:

1. for all $\ell' \in \mathsf{lab}'(\mathcal{O})$ there is an $\mathcal{R} \in \mathsf{Rep}(\mathcal{O}, \mathsf{lab}, c)$ such that $\mathcal{O}_{\ell'}^{\mathsf{lab}'} = \mathcal{R}$ and $\mathcal{R}$ is maximal; and
2. for every maximal $\mathcal{R} \in \mathsf{Rep}(\mathcal{O}, \mathsf{lab}, c)$ there exists some $\ell' \in \mathsf{lab}'(\mathcal{O})$ such that $\mathcal{O}_{\ell'}^{\mathsf{lab}'} = \mathcal{R}$.

The condition of maximality in this definition is used to guarantee that the new labeled ontology is irredundant. Notice that the maximal elements $\mathcal{R} \in \mathsf{Rep}(\mathcal{O}, \mathsf{lab}, c)$ are exactly those that one could consider solutions of an iterative contraction. For example, if $\mathcal{O}$ is the labeled ontology from Figure 2 (b), then $\mathsf{Rep}(\mathcal{O}, \mathsf{lab}, B \sqsubseteq E)$ contains the ontology $\{C \sqsubseteq D, D \sqsubseteq E, E \sqsubseteq F\}$, which is a repair of $\mathcal{O}_1^{\mathsf{lab}}$ w.r.t. $B \sqsubseteq E$. However, as this ontology is already contained in $\mathcal{O}_2^{\mathsf{lab}}$, it is not considered a solution of the contraction—and indeed, is not a maximal element of $\mathsf{Rep}(\mathcal{O}, \mathsf{lab}, B \sqsubseteq E)$.

An obvious approach for computing a labeled contraction is by expanding the ontology into all its contexts. Following Definition 6 literally, one can compute the set of repairs for each context w.r.t. $c$, remove all non-maximal ones, and encode the resulting set of ontologies with a new labeled ontology. Clearly, this naïve idea has all the drawbacks associated to storing all the solutions independently in the first place. Instead, we propose an optimized method that is based on manipulating the labels of the axioms directly.

Recall that the boundary $\nu$ of a labeled ontology $\mathcal{O}$ w.r.t. a consequence $c$ expresses all the contexts of $\mathcal{O}$ that entail $c$. Obviously, if we contract $c$ from any context that does not entail this consequence, then this context remains unchanged. Thus, we only need to change the contexts that appear in the boundary, which means modifying the labels of the axioms in $\mathcal{O}_\nu := \bigcup_{\ell \in \nu} \mathcal{O}_\ell^{\mathsf{lab}}$.

Let now $\ell \in \nu$. It is easy to see that for every repair $\mathcal{R}_\ell$ of $\mathcal{O}_\ell^{\mathsf{lab}}$ w.r.t. $c$, there is a repair $\mathcal{R}_\nu$ of $\mathcal{O}_\nu$ w.r.t. $c$ such that $\mathcal{R}_\ell = \mathcal{R}_\nu \cap \mathcal{O}_\ell^{\mathsf{lab}}$. Indeed, this is a direct consequence of the following facts: (i) $\mathcal{O}_\ell^{\mathsf{lab}} \subseteq \mathcal{O}_\nu$, and (ii) $\mathcal{R}_\ell \not\models c$. Thus, we can extend $\mathcal{R}_\ell$ with axioms from $\mathcal{O}_\nu \setminus \mathcal{O}_\ell^{\mathsf{lab}}$ until a repair of the larger set is found. This means that, if we find all the repairs of $\mathcal{O}_\nu$ w.r.t. $c$, then intersecting these with all the contexts in the boundary yields all the repairs of each of these contexts w.r.t. $c$. Notice that this approach might produce some redundant sub-ontologies: the same repair might be obtained from the intersection of two repairs with the same context, and some of these intersections might not be maximal, hence not being repairs. However, this redundancy can be easily checked through a set inclusion test.

Algorithm 1 performs labeled contraction following the ideas described above. It starts by computing the boundary $\nu$ of $\mathcal{O}$ w.r.t. $c$, and uses this boundary to find all the repairs of $\mathcal{O}_\nu$ w.r.t. $c$. These repairs are intersected with all contexts whose label is in $\nu$. All such intersections that are not sub-ontologies of some previously known solution to the contraction problem are stored in the set $\mathfrak{S}$. Then, the maximal sub-ontologies (w.r.t. set inclusion) from $\mathfrak{S}$ are identified as the sets $\mathcal{R}_1, \ldots, \mathcal{R}_n$. Finally, the labeling function $\mathsf{lab}$ is updated to remove the contexts in $\nu$ and add the newly found solutions $\mathcal{R}_i$.

**Algorithm 1** Context-based contraction

**Require:** Labelled ontology $\mathcal{O}$, consequence $c$
1: $\nu \leftarrow \mathsf{boundary}(\mathcal{O}, c)$
2: $\mathfrak{R} \leftarrow \mathsf{repairs}(\mathcal{O}_\nu, c)$
3: $\mathfrak{S} \leftarrow \emptyset$
4: **for** each $\mathcal{R} \in \mathfrak{R}$ **do**
5: **for** each $\ell \in \nu$ **do**
6: **if** $\mathcal{R} \cap \mathcal{O}_\ell^{\mathsf{lab}} \not\subseteq \mathcal{O}_m^{\mathsf{lab}}$ for all $m \notin \nu$ **then**
7: $\mathfrak{S} \leftarrow \mathfrak{S} \cup \{\mathcal{R} \cap \mathcal{O}_\ell^{\mathsf{lab}}\}$
8: **end if**
9: **end for**
10: **end for**
11: $\mathcal{R}_1, \ldots, \mathcal{R}_n \leftarrow \mathsf{maximal}(\mathfrak{S})$
12: $k \leftarrow \max \mathsf{lab}(\mathcal{O})$
13: **for** each $\alpha \in \mathcal{O}$ **do**
14: $\mathsf{lab}(\alpha) \leftarrow (\mathsf{lab}(\alpha) \setminus \nu) \cup \{k + i \mid \alpha \in \mathcal{R}_i, 1 \leq i \leq n\}$
15: **end for**
16: **return** $\mathsf{lab}$

For example, suppose that we want to retract the consequence $B \sqsubseteq E$ from the labeled ontology depicted in Figure 2 (b). In this case, we first compute the boundary $\nu = \{1, 5\}$, which yields $\mathcal{O}_\nu = \mathcal{O}_{\mathsf{ex}}$ (as defined in Example 3). The repairs of $\mathcal{O}_\nu$ w.r.t. $B \sqsubseteq E$ are precisely $\mathcal{R}_2$, $\mathcal{R}_3$, and $\mathcal{R}_4$ as depicted in Figure 1. Since these are already contained in some contexts not in $\nu$, $\mathfrak{S}$ remains empty, and the new labeling function simply removes the contexts 1 and 5 from the labeled ontology, as expected.

It is worth considering the steps 6 to 8 of Algorithm 1. As mentioned before, this test is used to guarantee that we do not include in $\mathfrak{S}$ any solution candidate that is redundant due to being contained in a previously known solution. As described in the algorithm, this would require comparing each intersection found with all the contexts not belonging to the boundary. Clearly, such a task would be extremely expensive. Fortunately, it is possible to exploit the labels of the ontology to optimize this test too. Recall that for every axiom $\alpha$, $\mathsf{lab}(\alpha)$ is the set of contexts to which $\alpha$ belongs. Thus, given a subontology $\mathcal{R}$, $\mathsf{con}(\mathcal{R}) := \bigcap_{\alpha \in \mathcal{R}} \mathsf{lab}(\alpha)$ yields the set of contexts that contain $\mathcal{R}$. To test the condition in the **if** statement from line 6, it then suffices to check that $\mathsf{con}(\mathcal{R}) \subseteq \nu$.

Now that we have seen how to contract a consequence from a labeled ontology, we turn our attention to the problem of expanding and revising an ontology with a new consequence.

## 5 Expansion and Revision

As it is the case in classical ontology update, the process of expanding a labeled ontology to include a consequence is relatively simple. By assumption, every consequence of the ontology language can also be expressed as an axiom. Thus, in

order to expand an ontology $\mathcal{O}$ to include the consequence $c$, it suffices to add the axiom $c$ to $\mathcal{O}$ in case that $\mathcal{O} \not\models c$, or leave $\mathcal{O}$ unchanged if $c$ is already entailed by this ontology. In the case of labeled expansion, we want to add the new axiom only to those contexts that do not entail $c$ already.

**Definition 7** (labeled expansion)**.** Let $\mathcal{O}$ be a labeled ontology with labeling function $\mathsf{lab}$, and $c \in \mathfrak{C}$. A *labeled expansion* of $\mathcal{O}$ w.r.t. $c$ is a labeled ontology $\mathcal{O}'$ with labeling function $\mathsf{lab}'$ such that $\mathsf{lab}(\mathcal{O}) = \mathsf{lab}'(\mathcal{O}')$ and for every $\ell \in \mathsf{lab}(\mathcal{O})$

$$(\mathcal{O}')^{\mathsf{lab}'}_{\ell} = \begin{cases} \mathcal{O}^{\mathsf{lab}}_{\ell} & \text{if } \mathcal{O}^{\mathsf{lab}}_{\ell} \models c \\ \mathcal{O}^{\mathsf{lab}}_{\ell} \cup \{c\} & \text{otherwise.} \end{cases}$$

As in the previous section, we can exploit the properties of the boundary to perform labeled expansion efficiently. The boundary allows us to identify the contexts that already entail $c$ from those who do not. After we have computed the boundary $\nu$ of $\mathcal{O}$ w.r.t. $c$, expansion consists simply of adding the axiom $c$ to $\mathcal{O}$ and labeling it with $\mathsf{lab}(\mathcal{O}) \setminus \nu$, unless $\nu = \mathsf{lab}(\mathcal{O})$, in which case nothing is done.

The case of revision is slightly more complex. Recall that revision can be seen as a two step operator in which a consequence $c$ is first expanded, and then inconsistency is contracted. However, the set of solutions of the contraction step is restricted to consider only those that entail the consequence $c$. To follow with this intuition, we assume that the set $\mathfrak{C}$ of consequences contains an element $\bot$ denoting that the ontology is inconsistent.

**Definition 8** (labeled revision)**.** Let $\mathcal{O}$ be a labeled ontology with labeling function $\mathsf{lab}$, and $c \in \mathfrak{C}$. A *labeled revision* of $\mathcal{O}$ w.r.t. $c$ is a labeled ontology $\mathcal{O}'$ with labeling function $\mathsf{lab}'$ such that for all $\ell \in \mathsf{lab}'(\mathcal{O}')$, $(\mathcal{O}')^{\mathsf{lab}'}_{\ell} \models c$, $(\mathcal{O}')^{\mathsf{lab}'}_{\ell} \not\models \bot$, and for each $\ell \in \mathsf{lab}(\mathcal{O})$ there is some $\ell' \in \mathsf{lab}'(\mathcal{O}')$ such that $(\mathcal{O}')^{\mathsf{lab}'}_{\ell'}$ is a revision of $\mathcal{O}^{\mathsf{lab}}_{\ell}$ w.r.t. $c$.

To perform such a labeled revision step, we combine the algorithm for performing expansion described above with the labeled contraction method described by Algorithm 1. The procedure obtained is described in Algorithm 2. As it can be easily seen, this algorithm follows the two step approach of first expanding $c$ and then contracting $\bot$. However, the **if** condition from line 13 requires additionally that the proposed solution entails the consequence that is being revised in the ontology; compare it with line 6 of Algorithm 1.

Notice that the correctness of Algorithm 2 w.r.t. revision requires that all the contexts in the input labeled ontology $\mathcal{O}$ are themselves consistent; otherwise, the resulting ontology might preserve some of these inconsistencies. This assumption is not really problematic for our approach. In fact, if revision is being used, it means that the application is expected to preserve consistency throughout the update procedure. Moreover, if this was not the case, we could expand the procedure to first contract any inconsistency, before executing the revision (or any other) operator.

As it can be seen from these two sections, the different update operators can be implemented directly in the labeled ontologies through a manipulation of the labeling function. Thus, iterative ontology updates preserving the minimum change principle can be performed through updates in a labeled ontology. In the next section we describe further reasoning problems over labeled ontologies that will be useful for the iterative ontology update problem.

**Algorithm 2** Context-based revision

**Require:** Labelled ontology $\mathcal{O}$, consequence $c$
1: $\nu \leftarrow \mathsf{boundary}(\mathcal{O}, c)$
2: **if** $\nu = \mathsf{lab}(\mathcal{O})$ **then**
3: **return** $(\mathcal{O}, \mathsf{lab})$
4: **end if**
5: $\mu \leftarrow \mathsf{lab}(\mathcal{O}) \setminus \nu$
6: $\mathcal{O}' \leftarrow \mathcal{O} \cup \{c\}$
7: $\mathsf{lab}'(c) \leftarrow \mathsf{lab}(\mathcal{O}) \setminus \nu$
8: $\nu' \leftarrow \mathsf{boundary}(\mathcal{O}, \bot)$
9: $\mathfrak{R} \leftarrow \mathsf{repairs}(\mathcal{O}'_{\nu'}, \bot)$
10: $\mathfrak{S} \leftarrow \emptyset$
11: **for** each $\mathcal{R} \in \mathfrak{R}$ **do**
12: **for** each $\ell \in \nu'$ **do**
13: **if** $\mathcal{R} \cap (\mathcal{O}')^{\mathsf{lab}'}_{\ell} \not\subseteq (\mathcal{O}')^{\mathsf{lab}'}_{m}$ for every $m \notin \nu'$ and $\mathcal{R} \cap (\mathcal{O}')^{\mathsf{lab}'}_{\ell} \models c$ **then**
14: $\mathfrak{S} \leftarrow \mathfrak{S} \cup \{\mathcal{R} \cap (\mathcal{O}')^{\mathsf{lab}'}_{\ell}\}$
15: **end if**
16: **end for**
17: **end for**
18: $\mathcal{R}_1, \ldots, \mathcal{R}_n \leftarrow \mathsf{maximal}(\mathfrak{S})$
19: $k \leftarrow \max \mathsf{lab}(\mathcal{O})$
20: **for** each $\alpha \in \mathcal{O}'$ **do**
21: $\mathsf{lab}'(\alpha) \leftarrow (\mathsf{lab}'(\alpha) \setminus \nu') \cup \{k+i \mid \alpha \in \mathcal{R}_i, 1 \leq i \leq n\}$
22: **end for**
23: **return** $(\mathcal{O}', \mathsf{lab}')$

## 6 Reasoning over the Best Contexts

After all the updates have been performed, the final goal is still to obtain one solution ontology that will be output as the solution to the iterative update problem. This solution should be the *best* context of the labeled ontology, for some adequate notion of quality of ontologies. If our updates all consisted of contraction of consequences, then the best solutions will be those contexts that contain the most axioms; these are the contexts from which the least amount of axioms were removed to get rid of all the unwanted consequences. If, on the other hand, the chain of updates is composed by a mixure of contraction and expansion (or revision) operations, then the best solution should still have the most of the original axioms possible, but should also contain as little of the newly added axioms as possible. In other words, finding the contexts with the largest cardinality does not suffice anymore. For this case, we use a notion of *rank* of the axioms in an ontology.

In addition to the set of contexts to which it belongs, every axiom $\alpha$ is associated to a natural number $\mathsf{rk}(\alpha)$, called its *rank*, that expresses the age of the axiom in the ontology. At the beginning of the iterative update process, all axioms in the input ontology are assigned a rank of 1. Whenever a new axiom is added to the ontology, through an expansion or revision step, the rank of this axiom is larger than that of all the previously existing axioms. Using this rank, it is pos-

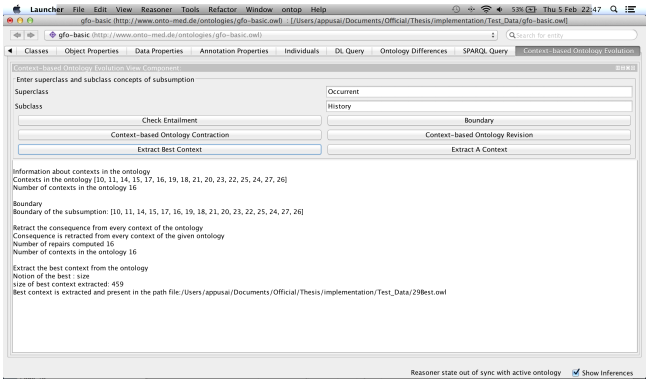


Figure 3: Screenshot of a Protégé plug-in for ontology update

sible to define different measures of quality for a context in an ontology. One can, for example, find the contexts whose maximum rank is the smallest possible. This is the notion that we choose, as it guarantees that later expansions did not modify the axiomatic structure of the solution ontology.

Before choosing the final answer to the update problem, a user might be interested in analysing the best solutions found so far according to their properties. For example, if after several contraction operations one is interested in obtaining an ontology that entails a consequence $c$, it might not be necessary to expand the ontology with $c$. Instead, one can search for one of the best solutions that already entails $c$, and provide it as a final answer. To help in this task, we generalize the notion of the boundary, to consider only the best solutions, for a given notion of best.

**Definition 9** (best boundary)**.** Let $\mathcal{O}$ be a labeled ontology and $c$ a consequence. The *best boundary* of $\mathcal{O}$ w.r.t. $c$ is the (unique) set $\nu \subseteq \mathsf{lab}(\mathcal{O})$ of labels such that $\ell \in \nu$ iff $\mathcal{O}_\ell^{\mathsf{lab}} \models c$ and $\mathcal{O}_\ell^{\mathsf{lab}}$ is one of the best contexts of $\mathcal{O}$.

Since the class of best contexts can be maintained during the whole ontology update process, finding the best boundary reduces simply to finding the boundary over a restricted class of labels. The existing algorithms for computing the boundary can thus be easily adapted to compute the best boundary, too.

In the next section, we describe a prototype implementation of these ideas for updating description logic ontologies and reasoning with the best solutions.

## 7 Implementation and Evaluation

All the algorithms described in this paper were implemented in a prototype plug-in of the ontology editor Protégé.[2] Figure 3 shows a screenshot of this plug-in. Our tool is an extension of the context-based reasoner CoBRA[3] that was originally developed for computing the boundary of a description logic labeled ontology w.r.t. a given consequence [Peñaloza and Thuluva, 2014]. CoBRA uses a black-box based approach and exploits the functionalities provided by highly optimized DL reasoners to provide answers efficiently. It is worth mentioning that, although the ideas developed and the black-box mechanism is independent of the specific ontology language used, and in particular of the DL chosen, the current implementation of CoBRA makes extensive use of the $\mathcal{EL}^{++}$ reasoner ELK [Kazakov, Krötzsch, and Simančík, 2014]; in particular of the incremental reasoning functionality that this reasoner provides. For this reason, the functionality of both, CoBRA and our plug-in, is currently limited to $\mathcal{EL}^{++}$ ontologies.

The computation of the boundary is based on the optimized HST method proposed by Baader, Knechtel, and Peñaloza [2012]. In this method, so-called MinAs [Baader and Peñaloza, 2010a; Baader and Peñaloza, 2010b; Kalyanpur et al., 2007] are used to restrict the search-space of the contexts that entail a consequence. To implement this method, CoBRA exploits the explanation functionality provided by the reasoner HermiT [Motik, Shearer, and Horrocks, 2009].[4] To communicate with all the reasoners and with Protégé, the tool uses the OWL API.

To reason over the best answers, the size of each context and the rank of the axioms are maintained in hash tables that allow for easy access. Whenever a reasoning task is required, the different labels, the axioms in the contexts they define, and their different properties can be retrieved easily. The plug-in receives as input an ontology, which could be labeled or not, and a subsumption relation of interest. The user can then choose to compute the boundary for the subsumption relation, to update the ontology either by retracting or revising the consequence, to find the best contexts of the ontology, or to extract one arbitrary context of the ontology. For the last two tasks, a subsumption relation can be specified so that only those solutions that entail it are considered.

We tested our plug-in using the $\mathcal{EL}^+$ version of GALEN and the 2010 version of SNOMED CT. Both of these ontologies were chosen by their size, and their expressivity, which can be handled by CoBRA. As our prototype is not fully optimized, and is based on black-box methods for handling the boundary and other intermediate steps, our experiments are more targeted to understanding the advantages of using our approach for iterative update, and identifying the bottlenecks in the efficiency of our implementation. To achieve these goals, each experiment was based on five distinct consequences of the input ontology. Four of these consequences were contracted from the ontology and the fifth was used to extract one best solution that entailed it. The size of this best solution was compared to the result of extracting one maximum repair after each contraction. Surprisingly for us, the results showed that preserving all the information during contraction did not provide a much better answer than extracting an ontology at every step. Indeed, in all the experiments performed on SNOMED, the size of the answers was the same, while for GALEN, at most one axiom was removed unnecessarily by the naïve approach. This might be explained by the fact that these ontologies are well-structured, and usually

[2] http://protege.stanford.edu/

[3] http://cobrareasoner.sourceforge.net/

[4] http://hermit-reasoner.com/

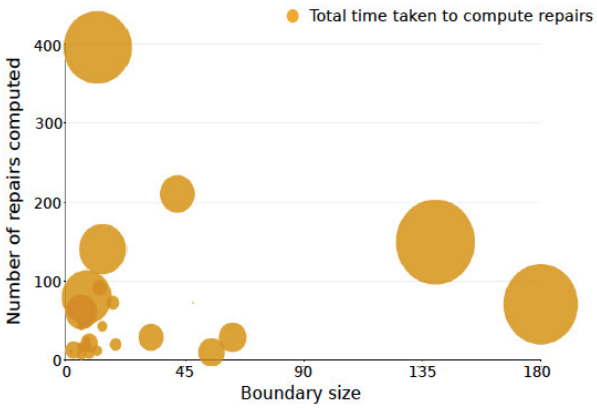


Figure 4: Total time for computing repairs in relation to the number of repairs and boundary size for GALEN

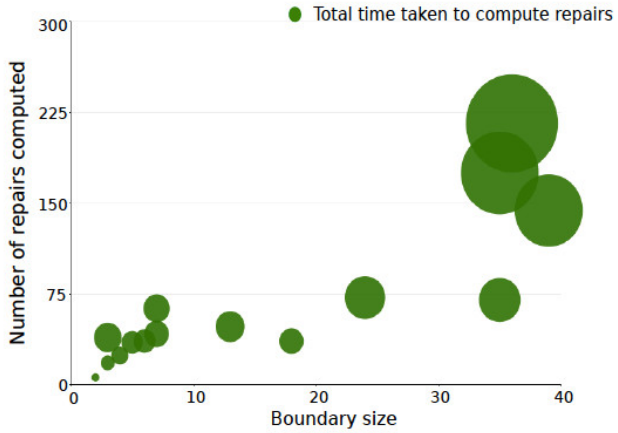


Figure 5: Total time for computing repairs in relation to the number of repairs and boundary size for SNOMED

consequences are caused by very small sub-sets of axioms.

More interesting is the analysis of the factors that reduce the efficiency of our implementation. Analysing the execution times of all the experiments made, we see that the main bottleneck in our tool is the time required to compute the repairs when a contraction is made. Moreover, the time required for computing these repairs is strongly correlated with the size of the boundary for the consequence (that is, the number of contexts that will need to be modified) and with the total number of repairs computed. This correlation can be easily visualized in Figures 4 (for GALEN) and 5 (SNOMED). We notice that these experiments still took over ten minutes to execute. This means that our prototype is still far from being at a production state. However, being based fully in black-box methods, it has still a large margin for optimizations, both in terms of the reasoners it calls, and in the integrations of these reasoners into the overall tool.

To reduce the time required to compute the repairs, we can then optimize the boundary computation algorithm. A further analysis of our executions shows that the MinA computation approach from HermiT results very inefficient when the ontology has many contexts. One possibility is then to find a better implementation of MinA computation to use within our tool. As soon as one is developed, our method will be immediately improved.

For a more detailed analysis of the methods, and more information about the experimental setting, we refer the interested reader to [Thuluva, 2015]. Obviously, more experiments and further optimizations are necessary before the tool can be effectively used for practical applications.

## 8 Conclusions

We have studied a new approach for performing iterative ontology update, preserving the principle of minimal change. Our approach is based on the idea of preserving many ontologies together, represented compactly as a single labeled ontology. Within this setting, we propose contraction, expansion and revision operators that work directly on the labeled ontology, and modifying the labels that define the contexts, or solutions.

Using off-the-shelf reasoning tools for description logics, we implemented a prototype iterative update tool that applies our methods. Although our prototype is still far from being effective for real-life applications, it shows that our ideas are feasible. Moreover, an experimental analysis shows the main components that affect the efficiency of our methods. As future work, we plan to further optimize and analyse our methods.

## Acknowledgements

This work was partially supported by the German Research Foundation (DFG) under the SFB 912 'HAEC' and the Cluster of Excellence 'cfAED,' and was developed while the first author was still affiliated with TU Dresden and the Center for Advancing Electronics Dresden, Germany.

## References

[Alchourrón, Gärdenfors, and Makinson, 1985] Alchourrón, C. E.; Gärdenfors, P.; and Makinson, D. 1985. On the logic of theory change: Partial meet contraction and revision functions. *Journal Symbolic Logic* 50(2):510–530.

[Baader and Peñaloza, 2010a] Baader, F., and Peñaloza, R. 2010a. Automata-based axiom pinpointing. *Journal of Automated Reasoning* 45(2):91–129.

[Baader and Peñaloza, 2010b] Baader, F., and Peñaloza, R. 2010b. Axiom pinpointing in general tableaux. *Journal of Logic and Computation* 20(1):5–34.

[Baader et al., 2007] Baader, F.; Calvanese, D.; McGuinness, D. L.; Nardi, D.; and Patel-Schneider, P. F., eds. 2007. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2nd edition.

[Baader, Brandt, and Lutz, 2005] Baader, F.; Brandt, S.; and Lutz, C. 2005. Pushing the $\mathcal{EL}$ envelope. In *Proceedings of*

*the 19th International Joint Conference on Artificial Intelligence (IJCAI'05)*. Edinburgh, UK: Morgan Kaufmann.

[Baader, Knechtel, and Peñaloza, 2012] Baader, F.; Knechtel, M.; and Peñaloza, R. 2012. Context-dependent views to axioms and consequences of semantic web ontologies. *Journal of Web Semantics* 12–13:22–40. Available at http://dx.doi.org/10.1016/j.websem.2011.11.006.

[Baader, 2003] Baader, F. 2003. Terminological cycles in a description logic with existential restrictions. In Gottlob, G., and Walsh, T., eds., *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI'03)*, 325–330. Morgan Kaufmann.

[Ceylan and Peñaloza, 2014] Ceylan, I. I., and Peñaloza, R. 2014. The bayesian description logic bel. In *Proc. of IJCAR'14*, volume 8562 of *LNCS*. Springer.

[Darwiche and Pearl, 1997] Darwiche, A., and Pearl, J. 1997. On the logic of iterated belief revision. *Artificial Intelligence* 89(1-2):1–29.

[Kalyanpur et al., 2007] Kalyanpur, A.; Parsia, B.; Horridge, M.; and Sirin, E. 2007. Finding all justifications of OWL DL entailments. In *Proceedings of the 6th International Semantic Web Conference and 2nd Asian Semantic Web Conference, ISWC 2007, ASWC 2007*, volume 4825 of *Lecture Notes in Computer Science*, 267–280. Springer.

[Kazakov, Krötzsch, and Simančík, 2014] Kazakov, Y.; Krötzsch, M.; and Simančík, F. 2014. The incredible ELK: From polynomial procedures to efficient reasoning with $\mathcal{EL}$ ontologies. *Journal of Automated Reasoning* 53:1–61.

[Ludwig and Peñaloza, 2014] Ludwig, M., and Peñaloza, R. 2014. Error-tolerant reasoning in the description logic EL. In Fermé, E., and Leite, J., eds., *Proceedings of the 14th European Conference on Logics in Artificial Intelligence (JELIA'14)*, volume 8761 of *Lecture Notes in Artificial Intelligence*, 107–121. Madeira, Portugal: Springer-Verlag.

[Motik, Shearer, and Horrocks, 2009] Motik, B.; Shearer, R.; and Horrocks, I. 2009. Hypertableau Reasoning for Description Logics. *Journal of Artificial Intelligence Research* 36:165–228.

[Peñaloza and Thuluva, 2014] Peñaloza, R., and Thuluva, A. S. 2014. Cobra, a demo. In Keet, C. M., and Tamma, V., eds., *Proceedings of the 11th International Workshop on OWL: Experiences and Directions (OWLED 2014)*, volume 1265 of *CEUR Workshop Proceedings*.

[Reiter, 1987] Reiter, R. 1987. A theory of diagnosis from first principles. *Artificial Intelligence* 32(1):57–95.

[Thuluva, 2015] Thuluva, A. S. 2015. Iterative ontology update with minimum change. Master's thesis, Dresden University of Technology, Germany.