Fakultät Informatik

EMCL Master Thesis

Correcting Access Restrictions: a Range-based Approach

by

Eldora
born on February 20^{th}, 1985, Jakarta, Indonesia

Supervisor : **Prof. Dr.-Ing. Franz Baader**
Advisors : **Dr.-Ing. Martin Knechtel,**
Dr. rer. nat. Rafael Peñaloza Nyssen

To my parents and my sister.

Declaration

Author:	**Eldora**
Student ID Nr.:	**3552563**
Title:	**Correcting Access Restrictions: a Range-based Approach**
Degree:	**Master of Science**
Date of Submission:	**28/02/2011**

Hereby I certify that the thesis has been written by me. Any help that I have received in my research work has been acknowledged. Additionally, I certify that I have not used any auxiliary sources and literature except those I cited in the thesis.

Signature of Author

Abstract

Ontologies, as understood in Knowledge Representation area, are conceived as a means to represent conceptual knowledge. An ontology is typically shared between a number of users. When sharing an ontology, it is common that each user has different views, i.e. the sub-ontology that is accessible to him. However, specially for large-scale ontologies, it is a difficult task to predict which users will be able to deduce a consequence from sub-ontology they can access. A lattice-based approach has been developed for solving this problem, by computing an access label for every ontological consequence. If the knowledge engineer wants then to modify the access label of one of these consequences (e.g. make a consequence unavailable to some users), then he must find the axioms that need to be relabeled. In this work we present algorithms for finding a set of these axioms with minimal cardinality such that, when relabeled to a specified element of the lattice grant or remove access of a set of users to a consequence. These algorithms deal with some range-based access label modifications. The implementation of the algorithms is using programming language Java with Pellet as the OWL reasoner. An empirical evaluation is performed on two large-scale ontologies with different expressiveness, i.e. SNOMED CT [sno] and FUNCT [GKL09].

Contents

List of Figures

List of Tables

List of Symbols and Abbreviations

Symbol	Description	Definition
$\bigoplus_{\ell \in K} \ell$	least upper bound (join) of K	page 7
$\mu_{\mathcal{S}}$	join of labels of $\mathcal{S}$	page 8
$\bigotimes_{\ell \in K} \ell$	greatest lower bound (meet) of K	page 7
$\lambda_{\mathcal{S}}$	meet of labels of $\mathcal{S}$	page 8
$\odot$	normal termination	page 11
$\diamondsuit$	early path termination	page 11

Abbreviation	Description	Definition
CS	change set	page 15
DL	Description Logic	page 1
HST	Hitting Set Tree	page 9
IAS	minimal inserted axiom set	page 16
MinA	minimal axioms set	page 6
MinLab	minimal label set	page 11
NGEQ	not greater than or equal to	page 26
NLEQ	not less than or equal to	page 28
RAS	minimal removed axiom set	page 16

Chapter 1

Introduction

In many aspects in the world, such as economics, politics, and/or education, information is essential. Information should be received and shared correctly. It must be only accessed by the right individuals. To ensure that the flow of information from one user to another is correct, a system is built for information with predefined access restriction schemes. Such a scheme defines a particular part of the information that can be accessed publicly available or restricted to particular individuals.

The information can be represented in a logic-based way, particularly in description logics (DLs). The well known usage of DL is to provide formalism on representing knowledge of ontologies. An ontology [BLM+05] is a description of a domain. Ontologies use description logic as a powerful language to describe them. DL has been employed in various domains of knowledge, such as system engineering, bio-medical informatics, and language processing ontologies.

The size of the ontologies varies from small to big ones. A small ontology can be built and checked easily if it is presenting the right knowledge. But, an ontology, especially a big one, can have unsatisfiable concepts which are not explicit. In [SC03], Schlobach and Cornet mentioned that in order to build a coherent ontology, one needs to identify the unsatisfiable concepts. This means that one needs to know what the reason for a particular consequence holds. This is often an expensive task, especially in a large scale ontology. To deal this task, they investigated how to compute the relevant axioms to the incoherence and named the task to compute it *axiom pinpointing*.

Imagine the following scenario. There is a knowledge engineer who manages the ontology and defines the privacy of knowledge in the ontology, i.e. some part of the ontology might or not be accessed, e.g. for reading, writing, etc. He is assigned to manage a large ontology. In this ontology, there are some particular consequences that should not be accessible by some users. In order

to ensure the security of the consequences, the knowledge engineer specifies the access restrictions. It is intended to give a level of privilege to users. In the other words, he gives a user an access to a particular sub-ontology.

From the sub-ontology which is available to the user, the user can deduce the consequences that can be derived from it. To ensure that a user can only deduce the particular consequence of the ontology which is accessible to her, one can provide many sub-ontologies to different users, but this is very expensive to build a lot of ontologies. Another way is that one can build a framework to restrict reading access to an ontology as proposed in [BKP09]. The idea of the framework is to compute a specific label that determines whether users can deduce the consequence or not. Each user is given a label that defines an access right or view for the user, i.e. the view of the ontology which is available for this user. The view is a subset of the ontology (sub-ontology) which is selected by a particular criterion.

To make this idea worked, the axioms of the ontology are given specific labels, which are structured in a lattice and are representing access restrictions. By the labeling of the axioms, every consequence in the ontology implicitly inherits a label, which is named as a boundary. These labels determine if a user labeled with a label ℓ can deduce the consequences or not. In other words, if this user can see some sub-ontologies, then the implicit knowledge, i.e. the consequences, from the sub-ontologies are deducible to this user.

The authors proposed that labels are defined in a lattice L whose elements stand for a specific access view. The decision to use a lattice is that a lattice gives more flexibility to present the real world implementations. It represents not only hierarchical levels, but also incomparable levels.

The question is how to know if a consequence still follows from the ontology w.r.t. the lattice or how to know if the user with label ℓ can deduce a particular consequence. The answer is on how the semantic of the lattice is defined and what the minimal axioms sets that can deduce if the consequence follows from the ontology are. If the minimal axioms sets are known, we can easily compute the label which corresponds to the consequence w.r.t. the lattice. But, often they are not known or too many to be computed manually.

The solution introduced in [BKP09] by Knechtel and Peñaloza is delivered by using some black box approaches. One of them is axiom pinpointing. The axiom pinpointing computes the smallest label of each axiom in every minimal axioms set and later it computes the biggest one out from the smallest labels. The result is the label of the consequence. Once the consequence label is obtained, we can see which the users can deduce it.

In some cases, axioms can entail a consequence which is considered private accessible by a user who is supposedly not able to see it. Some public axioms might possibly be labeled by private labels or the other way around. For a large

scale ontology, correcting this flaw is a difficult task. One way to solve this task is to reset the computed the label which determines the accessibility of a given consequence which can be done by relabeling some axioms. To do this repairing, the knowledge engineer might want a suggestion informing which axioms need their labels to be changed.

Knechtel and Peñaloza in [KP10a, KP10b] have introduced a way of getting the set of axioms that needs to be relabeled. In those papers, they present some methods in order to obtain the new label which determines the accessibility of a given consequence equal to a given goal label. By assuming that the knowledge engineer knows the exact label ℓ for determining the consequence c, they propose a set $\mathcal{S}$ of axioms of minimal cardinality such that if all members of $\mathcal{S}$ are labeled to the goal label ℓ_g, then the boundary equals to the goal label. The basic idea of their solution is by selecting axioms which are labeled smaller or greater than the goal label. If we want to change the boundary to a greater label than what we have computed, then the selected axioms which we will relabel are the ones with labels smaller than the label. And if we want the consequence label to be a value that is smaller than the computed label, then we select axioms which have greater labels than the value we want.

What has motivated us is to allow the label of the given consequence change in many conditions which is beyond equality to a given value by the knowledge engineer. That means we want to allow the change not to a specific value. In more concrete ways, the label which shows the restriction is allowed to be changed as follows: given a limit of which the target label may occur. This means not to specify the label on an exact label. Instead, we specify on an accepted range. We introduce four limitations or conditions: the label which determines the accessibility of the consequence should be less than or equal to ℓ_g, greater than or equal to ℓ_g, not less than or equal to ℓ_g, and not greater than or equal to ℓ_g. The intuition is to give flexibility to the knowledge engineer such that he can grant or remove rights to some users.

In this work, we present a solution to obtain a set of axioms whose labels need to be changed in order to satisfies the given condition. The solution is implemented using a hitting set tree algorithm which has been used for computing the label which determines the accessibility of the given consequence and for repairing it to a specific label. The algorithm is then implemented under Java programming language. It includes some algorithms to help the repairing done. Some of them are one to compute the boundary, one to compute the minimal axiom sets that explain why a consequence follows from an ontology, and one to compute the smallest set of axioms which need to be relabeled depending the given conditions. The solution for some conditions is using three nested hitting set tree algorithms. Each of the hitting set tree algorithms has an auxiliary procedure which is used to computed a single particular set. Using Pellet as

the reasoner, the implementation is able to provide the minimal solution, which we evaluate by checking if we obtain the satisfying boundary after changing the labels of the solution. We build an interface also in Java programming language to provide easy usage to this boundary changes. We evaluate the implementation using two ontologies and five labeling lattices. We also provide results from the evaluation.

The work is distributed as follows. Chapter 2 contains basic concepts and definitions that are used for the rest of the work. Previous related work is recalled for the sake of completeness. This chapter includes the general notations, the computation of the consequence label, and the repairing solution from the one we have mentioned. Chapter 3 describes theories and methods that build the solution to answer the purpose of this work, i.e. to allow general range based condition to the repair of the boundary. Examples are included in Chapters 2 and 3. Chapter 4 describes how the preparation of the test case and the evaluation of the implementation are conducted. The results are discussed and presented in tables and charts. The last chapter concludes the work and suggests some ideas for future work.

Chapter 2

Theoretical Background

This chapter introduces basic concepts and results that will be used for the rest of this work, recalled from [BKP09, KP10a, KP10b].

2.1 Ontologies

Description logics (DLs) [BCM+03] are a family of knowledge representation languages. Properties of DLs are the well structured syntax and well understandable semantics for representing the knowledge of an application domain. Description logics are good candidates for an ontology language which requires a well-designed and well-defined language, and also the availability of powerful reasoning tools. Nevertheless, we do not specify any ontology language in this work. The reason is to get the most general results that can be instantiated to any case.

"An ontology is a formal, explicit specification of a shared conceptualisation." is a definition that given in [SBF98]. It contains concepts of one or some specific domains that has to be machine readable. In order to represent an ontology in a computer, it needs an ontology language to describe it. An ontology language is a formal language to construct the ontology. Description logic is one of formal languages that is often used to describe and to represent ontologies.

In the following we describe general terminology of DL used in this thesis to describe and define ontologies.

A *consequence* is a derivable knowledge from a set of given knowledges. A *monotone consequence relation* $\models$ for a given ontology language is a binary relation between ontologies $\mathcal{O}$ and a consequence c such that if $\mathcal{O} \models c$, then for any $\mathcal{O}' \supseteq \mathcal{O}$, $\mathcal{O}' \models c$. We say that the ontology $\mathcal{O}$ entails the consequence c (c follows $\mathcal{O}$) if $\mathcal{O} \models c$ and we also say that the ontology $\mathcal{O}$ does not entail the consequence c if $\mathcal{O} \not\models c$.

For a given consequence c of the ontology $\mathcal{O}$ where $\mathcal{O} \models c$, there is a minimal set of axioms that makes the relation true. This set is called a minimal axioms set (*MinA*). We can also see this in the opposite direction. For $\mathcal{O} \models c$, there is a minimal set of axioms that by removing it from the ontology makes the consequence relation false. A set of axioms which are contained in $\mathcal{O}$, if removed from the ontology, falsifies $\mathcal{O} \models c$ is called a *diagnosis* [Rei87, KP10b, KP10a, Rym92]. How to obtain this set is called the diagnostic problem.

Definition 2.1 (MinA and Diagnosis [BKP09, KP10b, KP10a])**.** *Let $\mathcal{O}$ be an ontology and c a consequence following from $\mathcal{O}$. A* MinA *is a sub-ontology $\mathcal{S} \subseteq \mathcal{O}$ such that $\mathcal{S} \models c$ and that every $\mathcal{S}' \subset \mathcal{S}$, $\mathcal{S}' \not\models c$.*

A diagnosis *for $\mathcal{O}$ and c is a sub-ontology $\mathcal{S} \subseteq \mathcal{O}$ such that $\mathcal{O} \setminus \mathcal{S} \not\models c$ and for all $\mathcal{S}' \subset \mathcal{S}$, $\mathcal{O} \setminus \mathcal{S}' \models c$.*

The following example presents an ontology describing a country.

Example 2.2. *Georgia is situated in both Eastern Europe and Western Asia. The following axioms are in the ontology $\mathcal{O}_{Georgia}$.*

$$
\begin{aligned}
&o_1 : Georgia \sqsubseteq EasternEurope \sqcap WesternAsia,\\
&o_2 : EasternEurope \sqsubseteq Europe \sqcap Asia,\\
&o_3 : WesternAsia \sqsubseteq Europe \sqcap Asia,\\
&o_4 : Europe \sqsubseteq World,\\
&o_5 : Asia \sqsubseteq World.
\end{aligned}
$$

There are five facts which are axioms $o_1, \ldots, o_5$. Some of the implicit consequences that can be extracted from the ontology are $Georgia \sqsubseteq Europe$, $Georgia \sqsubseteq Asia$, $Georgia \sqsubseteq World$, $EasternEurope \sqsubseteq World$, and $WesternAsia \sqsubseteq World$.

If we want to know why the ontology $\mathcal{O}_{Georgia}$ in Example 2.2 entails $Georgia \sqsubseteq World$, we can compute a MinA. It has four MinAs and three diagnoses for this consequence. The MinAs are $\{o_1, o_2, o_4\}$, $\{o_1, o_2, o_5\}$, $\{o_1, o_3, o_4\}$, and $\{o_1, o_3, o_5\}$. The diagnoses for the ontology $\mathcal{O}_{Georgia}$ and the consequence $Georgia \sqsubseteq World$ are $\{o_1\}$, $\{o_2, o_3\}$, and $\{o_4, o_5\}$.

In a large scale ontology often that the implicit consequences are supposed to be hidden or not accessible for some users. One way to do so is to create several sub-ontology for different users. This is an expensive solution which can end up with exponantially many different ontologies. Another solution presented in [BKP09] is by giving labels to the axioms of ontology, such that the label of the axiom determines if the axiom is contained in the sub-ontology which are accessible from a particular user criterion. The following section shows the labeling lattice and its terminology.

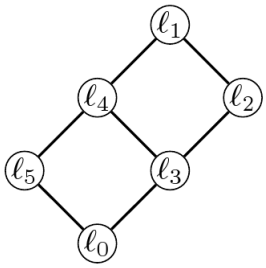


Figure 2.1: A lattice $(L_d, \leq_d)$ [BKP09].

2.2 Labeling Lattice

The idea proposed in [BKP09] to give access restriction on an ontology is by attaching a label to each axiom of the ontology. The set of these labels is structured in a lattice. It is called a labeling lattice $(L, \leq)$ as proposed in [BKP09].* The *lattice* $(L, \leq)$ is a set L of labels together with a partial order $\leq$ such that every finite set of labels has a least upper bound (join) and a greatest lower bound (meet) w.r.t. $\leq$. The partial order $\leq$ has been chosen for a better representation of hierarchical access restrictions which is more relevant to real world usage. Given a $K \subseteq L$, we denote the join of K as $\bigoplus_{\ell \in K} \ell$ and the meet of K as $\bigotimes_{\ell \in K} \ell$. We use only the bounded lattices as the labeling lattice, such that it has a greatest and a least elements, denoted by ℓ_1 and ℓ_0. And for the subset of the lattice, i.e. an empty set, the join of it is ℓ_0 and the meet of it is ℓ_1.

Figure 2.1 shows a lattice with 6 elements where ℓ_1 is the greatest element. The edges show the relation of two nodes. Given two connected nodes of labels (ℓ_1, ℓ_2), the label ℓ_1 is greater than the label ℓ_2. The order of the lattice is defined by the position of the nodes. A greater label means that label ℓ_1 is accessible more publicly than label ℓ_2 w.r.t. the ontology. In the prespective of a user, a user with label ℓ_1 has less access to the ontology than a user with label ℓ_2. This is clear by the definition of the sub-ontology for every label mentioned below. For any two nodes which are not connected, the relation between them is incomparable.

For an ontology $\mathcal{O}$ and a lattice $(L, \leq)$, a *labeled ontology* is an ontology $\mathcal{O}$ whose axioms are labeled using a labeling function $\mathsf{lab} : \mathcal{O} \to L$. The set of all labels that occur in an ontology $\mathcal{O}$ is denoted by

$$L_{\mathsf{lab}(\mathcal{O})} := \{\mathsf{lab}(o) \mid o \in \mathcal{O}\}.$$

Every label $\ell \in L$ defines the sub-ontology

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

*A detailed introduction to lattices can be found in [DP02]

The sub-ontologies $\mathcal{O}_{=}$, $\mathcal{O}_{\leq}$, $\mathcal{O}_{\neq}$, $\mathcal{O}_{\not\geq}$, $\mathcal{O}_{\not\leq}$ are defined analogously. Given a set of labels L', the sub-ontology is defined as follows.

$$\mathcal{O}_{L'} := \{a \in \mathcal{O}_{=\ell} \mid \ell \in L'\}.$$

Given a sub-ontology $\mathcal{S} \subseteq \mathcal{O}$, the join and the meet of labels that occur in $\mathcal{S}$ are denoted by

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

An element $\ell \in L$ is called *join prime relative* to L_{lab} if for every $K_1, K_2, \ldots, K_n \subseteq L_{\mathsf{lab}}$ $\ell \leq \bigoplus_{i=1}^{n} \lambda_{K_i}$ implies that there is i, $1 \leq i \leq n$ such that $\ell \leq \lambda_{K_i}$. A user is labeled to a label that is a join prime relative to L_{lab} (is called *user label*). The set of all the user labels is denoted by U.

2.3 Boundary

In order to decide whether a user with a label ℓ can deduce a consequence c of an ontology $\mathcal{O}$, [BKP09] introduced a solution by computing a label for the consequence w.r.t. the labeling lattice and the ontology. They use a labeling lattice to structure labels that determine if the user is able to see the consequence of the given ontology or not. This label is defined as follows.

Definition 2.3 (Boundary [BKP09])**.** *Let $\mathcal{O}$ be an ontology and c a consequence. An element $\nu \in L$ is called a* boundary *for $\mathcal{O}$ and c if for every element $\ell \in L$ that is join prime relative to L_{lab} ℓ it holds that $\ell \leq \nu$ iff $\mathcal{O}_{\geq \ell} \models c$.*

A boundary, intuitively, divides the user labels $\ell \in U$ according the entailment of c from $\mathcal{O}_{\geq \ell}$. If $\ell \leq \nu$ for some boundary ν of a consequence c, then we know that the consequence c is available to the user with label ℓ. In other words, it divides the ontology into parts where the consequence is deducible and where it is not.

Given a labeled ontology, one can compute the label of a consequence. The following lemma in [KP10a, KP10b] is related to how a boundary can be computed.

Lemma 2.4. *If $\mathcal{S}_1, \ldots, \mathcal{S}_n$ are all MinA for $\mathcal{O}$, c, then $\bigoplus_{i=1}^{n} \lambda_{\mathcal{S}_i}$ is a boundary for $\mathcal{O}$, c. 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.*

The following example illustrates how the Lemma 2.4 is applied to get a boundary for a given consequence.

Example 2.5. *Given $\mathcal{O}_{Georgia}$ as the ontology and lattice $(L_d, \leq_d)$, we want to compute the boundary for the consequence $Georgia \sqsubseteq World$. The ontology is labeled by elements of the lattice $(L_d, \leq_d)$ under the $\mathsf{lab}(o_i) = \ell_i$ function.*

According to Lemma 2.4, we compute all MinAs in order to obtain a boundary. The boundary is computed as follows. From each MinA, we compute the meet of the labels of the axioms. The $\lambda_{\{o_1,o_2,o_4\}}$ is ℓ_3 which is computed by taking the label of each axiom and getting the infimum of them, i.e. $\ell_1 \otimes \ell_2 \otimes \ell_4$. For all MinAs $\{S_1, \ldots, S_n\}$, we compute the join of $\forall_{i=1,\ldots,n}\lambda_{S_i}$. The complete computation is as follows.

$$\lambda_{\{o_1,o_2,o_4\}} \oplus \lambda_{\{o_1,o_2,o_5\}} \oplus \lambda_{\{o_1,o_3,o_4\}} \oplus \lambda_{\{o_1,o_3,o_5\}} = \ell_3 \oplus \ell_0 \oplus \ell_3 \oplus \ell_0 = \ell_3.$$

We can also compute the boundary using the dual notion of MinA, i.e. diagnosis. According to Lemma 2.4, a boundary is obtained by computing the meet of all μ of diagnoses and μ of a diagnosis is obtained by computing the join of all labels of axioms in the diagnosis. The complete computation is as follows.

$$\mu_{\{o_1\}} \otimes \mu_{\{o_2,o_3\}} \otimes \mu_{\{o_4,o_5\}} = \ell_1 \otimes \ell_2 \otimes \ell_4 = \ell_3.$$

The set of user labels U according to the lattice is $\{\ell_2, \ell_3, \ell_5, \ell_0\}$, because these labels are join prime relative to the set of labels that appear in the ontology. Therefore, only users with label ℓ_3 or ℓ_0 are able to see the consequence $Georgia \sqsubseteq World$, because they are less than or equal to ℓ_3.

2.4 Label Optimized Axiom Pinpointing

Lemma 2.4 shows that in order to compute a boundary, it is sufficient to have all MinAs or all diagnoses. A way to obtain all MinAs is through axiom pinpointing. This term was introduced by Schlobach and Cornet in [SC03]. There are many constructions of axiom pinpointing shown in [SC03, BPS07, BP10a, BP10b].

Two black-box approaches to obtain a boundary for an ontology $\mathcal{O}$ and a consequence c were presented in [BKP09]. The approaches are the full axiom pinpointing and the label-optimized axiom pinpointing. The full axiom pinpointing approach based on Lemma 2.4 computes all MinAs without using the labeled ontology. It is optimized by considering and using the labeled ontology. Since, usually, the number of the labels is much smaller than the number of the axioms in the ontology, the label-optimized axiom pinpointing reduced a lot of the search space of the full axiom pinpointing approach. This optimized approach computes minimal label sets, which we describe later on this section.

The label-optimized axiom pinpointing approach [BKP09] is a variant of Hitting-Set-Tree method (HST approach). The HST approach was first introduced by Reiter in [Rei87] to solve the diagnostic problem. The problem of computing all MinAs can be seen as the diagnostic problem which was addressed

Algorithm 1: Compute a minimal label set.

Procedure min-lab($\mathcal{O}, c$)
Input: $\mathcal{O}$: ontology; c: consequence
Output: $M_L \subseteq L$: a minimal label set for $\mathcal{O}$ and c

1: **if** $\mathcal{O} \not\models c$ **then**
2: **return** no minimal label set
3: $\mathcal{S} := \mathcal{O}$
4: $M_L := \emptyset$
5: **foreach** $k \in L_{lab}$ **do**
6: **if** $\bigotimes_{l \in M_L} l \not\leq k$ **then**
7: **if** $\mathcal{S}_{\neq k} \models c$ **then**
8: $\mathcal{S} := \mathcal{S}_{\neq k}$
9: **else**
10: $M_L := (M_L \setminus \{l | k < l\}) \cup k$
11: **return** M_L

by Reiter. It is a problem to reason which components in a system that are not functioning well. The system is described in a universal set and a set of conflict sets, where a conflict set is a subset of the universal set which is responsible for the problem. For given D is a collection of sets, a hitting set for D is a set $H \subseteq \bigcup_{S \in D} S$ such that $H \cap S \neq \emptyset$ for each $S \in D$. A MinA here can be assumed as a conflict set as in [Rei87] and a diagnosis can be seen as a hitting set according to the conflict set.

Originally, the hitting set tree algorithm was used to solve a problem where the solutions are in the number of the power set. It computes all the hitting sets. The problem to compute a boundary in [BKP09] can be handled using the hitting set tree algorithm with an arbitrary algorithm to compute a MinA. In the full axiom pinpointing, the algorithm computes all the MinAs. Each time a new MinA is computed, a variable is updated for the computation of the boundary. It basically computes a new MinA for each node which ensures that there are no two same MinAs. On each node, the algorithm continues to compute a MinA from a sub-ontology where it is a reduced one axiom from the ontology used in the previous computation of the MinA.

The problem discussed in [BKP09] was how to obtain the boundary for a given ontology and consequence. The diagnostic problem of Reiter was focused on obtaining the hitting sets of the given conflict sets. In [BKP09], they want to obtain a boundary which can be computed w.r.t. the ontology and the consequence if the MinAs have been known. Instead of precomputed and given, we use the hitting set tree to compute these MinAs and at the same time we compute the boundary as provided by Baader *et al.* [BKP09].

For the label-optimized version, a labeled ontology is used and they modified

Algorithm 2: Hitting set tree (HST) algorithm for computing the boundary

Procedure hst-boundary($\mathcal{O}, c$)
Input: $\mathcal{O}$: ontology; c: consequence
Output: boundary ν for c

1: **Global**: **C, H** $:= \emptyset; \nu$
2: $\mathcal{M} :=$ min-lab($\mathcal{O}, c$)
3: $C := \{\mathcal{M}\}$
4: $\nu := \bigotimes_{\ell \in \mathcal{M}} \ell$
5: **foreach** *label* $\ell \in \mathcal{M}$ **do**
6: expand-hst($\mathcal{O}_{\not\leq \ell}, c, \{\ell\}$)
7: **return** ν

Procedure expand-hst($\mathcal{O}, c, H$)
Input: $\mathcal{O}$: ontology; c: consequence; H: list of lattice elements
Output: boundary ν for c
Side effects: modifications to **C**, **H** and ν

1: **if** *there exists some* $H' \in \mathbf{H}$ such that $\{h \in H' | h \not\leq \nu\} \subseteq H$ or H' contains a prefix-path P with $\{h \in P | h \not\leq \nu\} = H$ **then**
2: **return** (early path termination $\diamondsuit$)
3: **if** *there exists some* $\mathcal{M} \in \mathbf{C}$ such that for all $\ell \in \mathcal{M}$, $h \in H$, $\ell \not\leq h$ and $\ell \not\leq \nu$ **then**
4: $\mathcal{M}' := \mathcal{M}$ (MinLab reuse)
5: **else**
6: $\mathcal{M}' :=$ min-lab($\mathcal{O}_{\not\leq \nu}, c$)
7: **if** $\mathcal{O}_{\not\leq \nu} \models c$ **then**
8: $\mathbf{C} := \mathbf{C} \cup \{\mathcal{M}'\}$
9: $\nu := \nu \bigoplus \otimes_{\ell \in \mathcal{M}'} \ell$
10: **foreach** *label* $\ell \in \mathcal{M}'$ **do**
11: expand-hst($\mathcal{O}_{\not\leq \ell}, c, H \cup \{\ell\}$)
12: **else**
13: $\mathbf{H} := \mathbf{H} \cup \{H\}$ (normal termination $\odot$)

HST of Reiter to compute a boundary by generating sets of labels of MinAs. The set is defined as follows.

Definition 2.6 (Minimal label set (MinLab) [BKP09])**.** *Given a labeled ontology $\mathcal{O}$ and a consequence c, a set $K \subseteq \{\mathsf{lab}(o) \mid o \in \mathcal{S}\}$ is called a* minimal label set *of $\mathcal{S}$ if the elements of K are incomparable and $\lambda_{\mathcal{S}} = \bigotimes_{\ell \in K} \ell$ where $\mathcal{S}$ is a MinA for $\mathcal{O}$ and c.*

Algorithm 1 generates a minimal label set by removing all labels which do not change the relation of the ontology and the consequence. The relation is that the ontology without axioms of these labels still entails the consequence. The algorithm returns variable M_L. The set M_L is a set of labels where axioms correspond to these labels cannot be removed from the ontology. At the end of the computation, M_L is the minimal label set. The procedure is as follows. First, by taking all labels used to label axioms in the ontology L_{lab}, it only considers label k that is smaller or incomparable to the meet of $M_L \bigotimes_{\ell \in M_L} \ell$. Any label k which is greater than the meet of M_L does not affect the result of the meet of

the set M_L, because the compution of a meet is to get the least element of all elements in M_L. Then, it checks whether the ontology without axioms labeled with k entails c or not. If the result is true, we remove all axioms which labeled with this label. If the result is false, it means that there are some axioms which explain why the consequence follows from the ontology. Therefore, the label k is added in the set M_L. The last, if there are labels in M_L which is smaller than k, then these labels are removed from M_L.

The HST Algorithm 2 computes the boundary for $\mathcal{O}$ and c. It builds the hitting set tree whose nodes are minimal label sets. Every node is labeled by a computed minimal label set from a call to the Algorithm 1. The first computed minimal label set (line 3) is the root of the hitting set tree. From the root node, the expansion of it depends on the number of elements of the set. If the set has three elements, then the node has three branches whose edges are labeled with each of the elements. Each branch represents a computation to get a new minimal label set. On each expansion branch, the computation uses different sub-ontology which is a reduced version of the previous used ontology, i.e. $\mathcal{O}_{\not\in \ell}$ where ℓ is the label of the edge. If there is no more minimal label set that can be computed, the branch is closed. In other word, it terminates the computation on this branch, returns to any unvisited branch, and tries to obtain a minimal label set from the correspond sub-ontology of the unvisited branch. Each time it obtains a new minimal label set, it updates the variable ν which by the end of the computation it contains the boundary, the variable $\mathbf{C}$ where all minimal label sets computed are stored, and the variable $\mathbf{H}$ that contains all the lists of labels from the root to the terminated branch.

The HST Algorithm 2 has some optimizations. First, there are two pruning rules that have been modified to handle this case better. These pruning rules are called the *early path termination* optimizations. Second, it generates nodes by depth-first search manner, which optimizes the MinLab reuse.

The first pruning rule has two conditions. The first is to close or to terminate the node n with path H such that there is a hitting set H' where $\{h \in H' \mid h \not\leq \nu\} \subseteq H$. At this point, only labels that are not less or equal to ν are needed to be checked whether they have been removed in other branch or not. If these labels exist in previous normal terminated branch, then node n which path H contains them is closed. The second condition is to close the node n with path $H = P$ where P is a prefix-path of some hitting set H'. The idea is that, when at some point we have removed the same labels as in a previous search, then we know that this point will result the same as the previous one and this point can be closed. The termination of this branch happens basically because we have searched for a MinLab on the previous branch, which has the same path on the current node. This also means that this branch will not give us a new MinLab if the computation is executed on this branch.

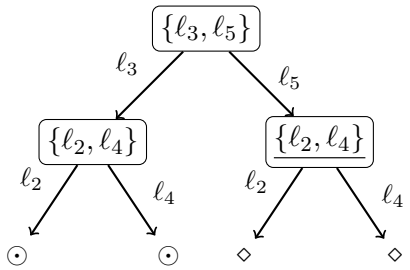


Figure 2.2: Expansion of HST modified version

Since this modified version is using depth-first search, the prefix-path P of a hitting set H' is then filtered by $\{h \in P \mid h \not\leq \nu\}$. The explanation is that once a minimal label set is found, then any next found minimal label set will be from $\mathcal{O}_{\not\leq \nu}$.

The second pruning rule is to reuse a computed minimal label set which has been computed in previous node. If there exist a minimal label set which follows the given conditions, then it is used as the current node. The conditions are for the current branch H, if there is a minimal label set $\mathcal{M} \in \mathbf{C}$ whose elements are not smaller than any element of H and the so far computed value of ν, then we use this set $\mathcal{M}$ as the new node for the current branch H. If there is no minimal label set which satisfies, then the Algorithm 2 calls the procedure $\mathsf{min\text{-}lab}(\mathcal{O}_{\not\leq \nu}, c)$.

Example 2.7. *Continuing the previous example, we now show how the HST modified algorithm works. We recall that there are four MinAs for the ontology $\mathcal{O}_{Georgia}$ and consequence $c = Georgia \sqsubseteq World$, namely $\{o_1, o_2, o_4\}$, $\{o_1, o_2, o_5\}$, $\{o_1, o_3, o_4\}$, and $\{o_1, o_3, o_5\}$. First, the* hst-boundary *calls* min-lab *which, we assume, returns $M_L = \{\ell_3, \ell_5\}$ which is the minimal label set of the MinA $\{o_1, o_3, o_5\}$. This M_L set is the root of the hitting set tree. At this point, $\nu = \ell_3 \otimes \ell_5 = \ell_0$. Then, it creates the first branch on the left by calling the* expand-hst *procedure which considers only $\mathcal{O}_{Georgia \not\leq \ell_3}$. The reason why only $\mathcal{O}_{georgia} \not\leq \ell_3$ is because on the edge ℓ_3 it will find a minimal label set from sub-ontology whose axioms is not labeled ℓ_3 and all labels smaller than ℓ_3. The sub-ontology $\mathcal{O}_{Georgia \not\leq \ell_3}$ entails c, so a new MinLab $\{\ell_2, \ell_4\}$ is obtained. We update $\nu = \ell_3$ from computing the join of the previous value of ν and the meet of new MinLab, i.e. $\ell_0 \bigoplus \{\ell_2 \otimes \ell_4\}$. Then the HST expands again with edge ℓ_2, this means the expansion consider axioms that are $\not\leq \ell_2$, i.e. only o_2. However, this ontology does not entail $Georgia \sqsubseteq World$ when we remove this axiom o_2. Therefore, this branch is terminated normally, which we denote by $\odot$. The same happens to the other branch with path $\{\ell_3, \ell_4\}$.*

Now we go back to the root and expand the root to the right side. The axioms in the original ontology are considered for the expansion except axioms which are $\not\leq \ell_5$, i.e. o_5 because it is the only axiom with a label $\not\leq \ell_5$. At this point, the early

Algorithm 3: Compute (partial) IAS

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

1: **Global** $l := 0$, n
2: **return** extract-partial-IAS-r($\mathcal{O}_{fix}, \mathcal{O}_{test}, c$)
Subprocedure extract−partial−IAS−r($\mathcal{O}_{fix}, \mathcal{O}_{test}, c$)
1: **if** $n = l$ **then**
2: **return** $\emptyset$
3: **if** $|\mathcal{O}_{test}| = 1$ **then**
4: $l := l + 1$
5: **return** $\mathcal{O}_{test}$
6: $\mathcal{S}_1, \mathcal{S}_2 :=$ halve($\mathcal{O}_{test}$) (partition $\mathcal{O}_{test}$ so that $||\mathcal{S}_1| - |\mathcal{S}_2|| \leq 1$)
7: **if** $\mathcal{O}_{fix} \cup \mathcal{S}_1 \models c$ **then**
8: **return** extract-partial-IAS-r($\mathcal{O}_{fix}, \mathcal{S}_1, c$)
9: **if** $\mathcal{O}_{fix} \cup \mathcal{S}_2 \models c$ **then**
10: **return** extract-partial-IAS-r($\mathcal{O}_{fix}, \mathcal{S}_2, c$)
11: $\mathcal{S}'_1 :=$ extract-partial-IAS-r($\mathcal{O}_{fix} \cup \mathcal{S}_2, \mathcal{S}_1, c$)
12: $\mathcal{S}'_2 :=$ extract-partial-IAS-r($\mathcal{O}_{fix} \cup \mathcal{S}'_1, \mathcal{S}_2, c$)
13: **return** $\mathcal{S}'_1 \cup \mathcal{S}'_2$

path termination conditions are not satisfied, but the MinLab reuse optimization is carried on. The variable $\boldsymbol{C}$ *contains* $\{\{\ell_3, \ell_5\}, \{\ell_2, \ell_4\}\}$. *The set* $\{\ell_3, \ell_5\}$ *does not satisfy the condition. The elements in the minimal label set* $\{\ell_2, \ell_4\}$ *are not less than or equal to* ν, *which is* ℓ_3, *and any* $h \in H$. *Therefore* $\{\ell_2, \ell_4\}$ *is reused to label this node. The MinLab reuse optimization is denoted by underlined node. From this node, the algorithm continues expanding it. On the left branch* ℓ_2, *the current path* H *satisfies the first condition, i.e. the current path* $H = \{\ell_5, \ell_2\}$ *is a superset of a set* $\{\ell_2\}$ *which is a previous hitting set* $\{\ell_3, \ell_2\}$ *and for every* $h \not\leq \nu$ *where* $h \in H$, h *is removed. The satisfying condition is* $\{\ell_2\} \subseteq \{\ell_5, \ell_2\}$. *Thus, the left branch is terminated. The algorithm expands the right branch of the* $\{\ell_2, \ell_4\}$ *node. The expansion with edge* ℓ_4 *is also terminated with an early termination condition. The condition* $\{\ell_4\} \subseteq H$ *where* $H = \{\ell_5, \ell_4\}$ *is satisfied. Therefore, all branches are terminated and the result of this algorithm is* $\nu = \ell_3$.

2.5 Repairing Ontology

For a given ontology $\mathcal{O}$ and a consequence c, once a boundary has been computed, [KP10b, KP10a] show that it is possible for the knowledge engineer to consider of changing the boundary to the condition that he wants. For example, a user u might be able to see the consequence c which is intended to be hidden from him. In both [KP10b, KP10a], Knechtel and Peñaloza show that a boundary can be

changed to a specific value (goal label) by computing a solution which is a set of axioms that have to be relabeled.

2.5.1 Modifying Boundary

To repair the boundary such that it follows the given condition, [KP10b, KP10a] fix a goal label ℓ_g and suggest the smallest set of axioms to be modified such that the boundary equals to ℓ_g . The following is the modified assignment for a given set.

Definition 2.8 ([KP10b, KP10a])**.** *Let $\mathcal{O}$ be an ontology,* 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*

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

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

The aim of their works is to provide a CS with minimum cardinality in order to perform as small as possible changes.

There are three cases in order to obtain the change set. They are $\ell_g < \nu$, $\ell_g > \nu$, and ℓ_g, ν are incomparable. For example, in our previous example, the computed boundary ν is ℓ_3. We want to change the boundary $\nu = \ell_0$. In this problem, we know that ℓ_g is smaller than ν. From Lemma 2.4, we can show that any diagnosis $\mathcal{S}$ is a CS. When all axioms in a diagnosis is relabeled to ℓ_g, the join of the diagnosis becomes equal to ℓ_g. Since all suprema of diagnoses are greater than or equal to ν, with the new join of that diagnosis we obtain a new boundary ℓ_g. Another example, we want to change the boundary ν to be equal to ℓ_4. We know that the goal value is greater than the boundary. From Lemma 2.4, we can derive that any MinA is a CS for this case. When all axioms in a MinA is relabeled to ℓ_g, the meet of the MinA becomes equal to ℓ_g. Since all infimua of the MinA are less than or equal to ν, with the new meet of that MinA we obtain a new boundary ℓ_g. Another example when we want to change the boundary to be ℓ_5. The goal value for this case is incomparable to the boundary. Therefore, the union of a MinA and a diagnosis is a CS for this case.

Example 2.9. *Recall the ontology $\mathcal{O}_{Georgia}$, consequence $Georgia \sqsubseteq World$, and the lattice L from previous examples. We already know that the boundary for $\mathcal{O}$, c is ℓ_3. Assume that we want the boundary to be ℓ_4. We are facing the case $\ell_g > \nu$. Thus, we take a MinA $\{o_1, o_2, o_4\}$. We relabel this MinA to ℓ_4. The meet of the MinA is now ℓ_4. Thus, together with the other infimum of the rest of MinAs, we obtain ℓ_4 as the supremum of them, i.e. the new boundary. However, $\{o_2\}$ is also a change set with smaller cardinality.*

Algorithm 4: Compute (partial) RAS

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

1: **Global** $l := 0$, $\mathcal{O}_{nonfix}$, n
2: **return** extract-partial-RAS-r($\emptyset, \mathcal{O}_{test}, c$)
Subprocedure extract−partial−RAS−r($\mathcal{O}_{hold}, \mathcal{O}_{test}, c$)
1: **if** $n = l$ **then**
2: **return** $\emptyset$
3: **if** $|\mathcal{O}_{test}| = 1$ **then**
4: $l := l + 1$
5: **return** $\mathcal{O}_{test}$
6: $\mathcal{S}_1, \mathcal{S}_2 :=$ halve($\mathcal{O}_{test}$) (partition $\mathcal{O}_{test}$ so that $||\mathcal{S}_1| - |\mathcal{S}_2|| \leq 1$)
7: **if** $\mathcal{O}_{nonfix} \setminus (\mathcal{O}_{hold} \cup \mathcal{S}_1) \not\models c$ **then**
8: **return** extract-partial-IAS-r($\mathcal{O}_{hold}, \mathcal{S}_1, c$)
9: **if** $\mathcal{O}_{nonfix} \setminus (\mathcal{O}_{hold} \cup \mathcal{S}_2) \not\models c$ **then**
10: **return** extract-partial-IAS-r($\mathcal{O}_{hold}, \mathcal{S}_2, c$)
11: $\mathcal{S}'_1 :=$ extract-partial-RAS-r($\mathcal{O}_{hold} \cup \mathcal{S}_2, \mathcal{S}_1, c$)
12: $\mathcal{S}'_2 :=$ extract-partial-RAS-r($\mathcal{O}_{hold} \cup \mathcal{S}'_1, \mathcal{S}_2, c$)
13: **return** $\mathcal{S}'_1 \cup \mathcal{S}'_2$

Nevertheless, either the smallest MinA, or the smallest diagnosis can be not the smallest change set, w.r.t. the cardinality of the set as shown in Example 2.9. The following definition generalizes a change set from a MinA or a diagnosis wchich contains only the required axioms.

Definition 2.10 (IAS and RAS [KP10b, KP10a])**.** *Let $\mathcal{O}$ be an ontology, c be a consequence, and ℓ_g be a goal label. 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 an IAS, a RAS, or the union of an IAS and a RAS when searching for change set with minimum cardinality.

Theorem 2.11 ([KP10a])**.** *Let ν be a boundary for $\mathcal{O}$, c, ℓ_g the goal label, m_R, m_I, and m_U the cardinalities of the smallest RAS, the smallest IAS, and the smallest union of an IAS and a RAS for ℓ_g, respectively. Then, for every IAS I and RAS R for ℓ_g it holds:*

- *if $\ell_g < \nu$ and $|R| = m_R$, then R is a CS of minimum cardinality,*
- *if $\nu < \ell_g$ and $|I| = m_I$, then I is a CS of minimum cardinality,*
- *if ν and ℓ_g are incomparable and $|I \cup R| = m_U$, then $I \cup R$ is a CS of minimum cardinality.*

Algorithm 5: Compute (partial) change set

Procedure extract−partial−CS$(\mathcal{O}, \mathsf{lab}, c, \ell_g, H, n)$

1: $\ell_c :=$ hst-boundary$(\mathcal{O}, c)$
2: **return** extract-partial-CS-aux$(\mathcal{O}, \mathsf{lab}, c, \ell_g,\ \ell_g \not\prec \ell_c \wedge \mathcal{O}_{\geq \ell_g} \not\models c,$
$\ell_g \not\succ \ell_c \wedge \mathcal{O}_{\not\leq \ell_g} \models c,\ H, n)$

Procedure extract−partial−CS−aux$(\mathcal{O}, \mathsf{lab}, c, \ell_g, \mathsf{is}_I, \mathsf{is}_R, H, n)$
Input: $\mathcal{O}, \mathsf{lab}$: labeled ontology; c: consequence; ℓ_g: goal label; is_I: decision to compute IAS; is_R: decision to compute RAS; H: HST edge labels; n: limit
Output: first n elements of a minimal CS $\mathcal{S} \subseteq \mathcal{O}$

1: **if** $1 \geq n$ *or* $\mathsf{is}_I \wedge \mathcal{O}_{\geq \ell_g} \cup (\mathcal{O}_{\not\leq \ell_g} \setminus H) \not\models c$ *or* $\mathsf{is}_R \wedge H \models c$ **then**
2: **return** $\emptyset$; `(HST normal termination)`
3: **if** is_I **then**
4: $I :=$ extract-partial-IAS$(\mathcal{O}_{\geq \ell_g}, \mathcal{O}_{\not\leq \ell_g} \setminus H, c, n)$
5: **if** is_R *and* $\mathcal{O}_{\not\leq \ell_g} \setminus I \models c$ **then**
6: $R :=$ extract-partial-RAS$(\mathcal{O}_{\not\leq \ell_g} \setminus I, \mathcal{O}_{\not\leq \ell_g} \setminus (I \cup H), c, n - |I|)$
7: **return** $I \cup R$

2.5.2 Computing the Smallest Change Set

In [KP10b], the authors describe an approach to compute the smallest IAS and RAS using axiom pinpointing, which is also optimized w.r.t. the labels of the axioms. The basic approach is based on the computation of MinAs and diagnoses, as shown by the following lemma.

Lemma 2.12. *Let I (R) be an IAS (RAS) for ℓ_g, then there is a MinA (diagnosis) $\mathcal{S}$ such that $I = \mathcal{S} \setminus \mathcal{O}_{\geq \ell_g}$ ($R = \mathcal{S} \setminus \mathcal{O}_{\leq \ell_g}$).*

The approach is improved by considering unchanged axioms of the ontology. Later, this is improved by only computing the partial IAS and RAS which reduces the search space and execution time. Finally, it is optimized by looking to the labels of IAS and RAS instead of looking to each axiom.

Change sets can be deduced from the set of all MinAs and diagnoses where an IAS or a RAS is a MinA or a diagnosis without the fixed elements, i.e. $\mathcal{O}_{\geq \ell_g}$ or $\mathcal{O}_{\leq \ell_g}$. To avoid unnecessary repetitions, we focus on computation of smallest IAS from here. The computation of smallest RAS can be treated in an analogous manner.

The basic approach to obtain all set of IAS is by computing all MinAs. The set of all MinAs is then filtered by removing all fixed elements of the ontology $\mathcal{O}_{\geq \ell_g}$. Finally, we search for an IAS with the smallest cardinality.

The goal is to find the smallest change set for changing the boundary. The previous approach can be improved by directly computing the set of IAS. Instead of using the auxiliary procedure of getting a MinA, a procedure to compute an IAS is introduced in Algorithm 3. A limit n is given to optimize the computation. It is used to ensure that each computation only computes an IAS with smaller

size of the smallest computed IAS so far. During the computation, it is updated to the size of smallest change set so far computed. When it finds an IAS of size at most n, it stops looking for more IAS because that shows that it will not find any smaller change set than the one with size n.

Algorithm 5 computes a minimal change set. It outputs a change set from the union of an IAS and RAS. This algorithm calls Algorithm 3 and Algorithm 4 where each of these algorithms computes partial IAS or RAS. Using the modified HST algorithm in [KP10b], it is claimed that the smallest change set can be found.

In the following chapter, we will discuss how to find a smallest change set for general cases, i.e. allowing boundary to be in a given criteria such as greater than or equal to a goal label.

Chapter 3

General Corrections of The Boundary

Previous work deals with the problem of allowing the boundary change to a specific goal label. In this chapter we show mainly our approach to deal with more general conditions for changing the boundary. As we have mentioned in Chapter 1, we want to allow general changes to the boundary. In specific way, we can change the boundary to a value less than or equal to, greater than or equal to, not less than or equal to, or not greater than or equal to some ℓ_g. We divide this problem into two sections. First, we will show an approach to obtain $\leq \ell_g$ and $\geq \ell_g$, called as positive conditions. Then, we will show approaches to solve $\not\leq \ell_g$ and $\not\geq \ell_g$, called as negative conditions.

Before going into the discussion, we revise the modified assigment and a change set $\mathcal{S}$ for these general cases. A *target label* (ℓ_t) is a label which is used to relabel the axioms.

Definition 3.1. *Let $\mathcal{O}$ be an ontology, c consequence* lab *a labeling function, $\mathcal{S} \subseteq \mathcal{O}$. The* modified assignment $\mathsf{lab}_{\mathcal{S},\ell}$ *where $\ell \in L$ and ℓ is a target label is given by*

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

Let C be a condition which is boundary $\leq$, $\geq$, $\not\leq$, or $\not\geq \ell_g$. A sub-ontology $\mathcal{S} \subseteq \mathcal{O}$ is called a change set *(CS) for C if the boundary for $\mathcal{O}$, c under the labeling function* $\mathsf{lab}_{\mathcal{S},\ell}$ *satisfies C.*

As we have seen in Chapter 2, [KP10b, KP10a] provide a suggestion of a change set to a knowledge engineer that wants to change the boundary for a given ontology $\mathcal{O}$ and a consequence c to a given goal label ℓ_g, with assumption that

he knows the computed boundary. It is often the case that a knowledge engineer gives a range of the "allowed" boundary. One of the reasons is simply because in a range he could have more flexibility in assigning the access restriction. For example, using the lattice $(L_d, \leq_d)$ from Figure 2.1, if the knowledge engineer wants to ensure that any user with labels ℓ_3 and ℓ_0 can deduce the consequence c, then the condition that suits this criterion is that the boundary should be greater than or equal to ℓ_3. With this condition, we ensure these users can deduce the consequence, but in the same hand we do not restrict the users with the other labels that could deduce the consequence if the boundary is ℓ_4, ℓ_2, or ℓ_1.

3.1 Positive Conditions

We now consider the case where the knowledge engineer gives a limitation such as a boundary should be greater than or equal to ℓ_g ($\nu \geq \ell_g$) or smaller than of equal to ℓ_g ($\nu \leq \ell_g$).

First, we focus on the condition greater than or equal to ℓ_g. It is obvious that for any computed boundary which is already satisfying the condition, we do not need to make any change on the ontology. On the contrary, if the previous boundary is not greater than or equal to ℓ_g, then we need to find a set of axioms such that changing the label of these axioms results in a boundary greater than or equal to ℓ_g. By Lemma 2.4, we know that a boundary is the supremum of the infima of the labels of all MinAs, i.e. it is greater than or equal to the infima of the labels of all MinAs. When we take a MinA and relabel its axioms to ℓ_g, automatically we change the infimum of this MinA to ℓ_g. The fact that we only execute the computation when the boundary does not satisfy the condition together gives information that the infimum ℓ_g is either greater to all the infima or incomparable to an infimum where the result of the join of all infima is a value greater than or equal to ℓ_g. In this way, after relabeling this MinA, we will obtain a new boundary which is equal to ℓ_g or greater than ℓ_g. The following lemma justifies it.

Lemma 3.2. *Let ℓ_g be a goal label, lab a labeling function, and $\mathcal{S}$ a MinA. $\mathcal{S}$ is a change set to repair a boundary to a value greater than or equal to ℓ_g by $\mathsf{lab}_{\mathcal{S},\ell_g}(\mathcal{O})$.*

Proof. Lemma 2.4 shows that for all MinAs $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_n$ for $\mathcal{O}$, c the boundary is $\bigoplus_{i=1}^{n} \lambda_{\mathcal{S}_i}$. We relabel all axioms in $\mathcal{S}$ to ℓ_g. The meet of the MinA $\mathcal{S}$ is now ℓ_g, i.e. $\lambda_{\mathcal{S}} = \ell_g$. Thus, the new boundary is greater than or equal to ℓ_g, because of the fact that a boundary ν is always greater than or equal to the meet of all MinAs. □

In the same fashion and argument, a diagnosis is a change set for changing a boundary to a label that is less than or equal to a goal value in the following lemma.

Lemma 3.3. *Let ℓ_g be a goal label, lab a labeling function, and $\mathcal{S}$ a diagnosis. $\mathcal{S}$ is a change set for changing a boundary to be less than or equal to ℓ_g by $\mathsf{lab}_{\mathcal{S},\ell_g}(o)$.*

Proof. Lemma 2.4 shows that if diagnoses $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_n$ for $\mathcal{O}$, c then $\bigotimes_{i=1}^{n} \mu_{\mathcal{S}_i}$ is a boundary for $\mathcal{O}$, c. We relabel all axioms in $\mathcal{S}$ to ℓ_g. The join of the diagnosis $\mathcal{S}$ is now ℓ_g, i.e. $\mu_{\mathcal{S}} = \ell_g$. Thus, the new boundary is less than or equal to ℓ_g. □

The MinA or the diagnosis which we choose as the change set might be not the minimal change set. This means that there are some axioms which can be removed from the change set. Refer to Lemma 2.4, a boundary is the least upper bound of all $\lambda_{\mathcal{S}}$ where $\mathcal{S}$ is a MinA. Changing the label of all the axioms in $\mathcal{S}$ is not necessary, because only some axioms with specific labels must be relabeled in order to change the meet of the MinA. Therefore, changing the label of some axioms in $\mathcal{S}$ is enough to repair the boundary. Only the axioms with label not greater or equal to ℓ_g need to be relabel.

Example 3.4. *Continuing Example 2.9, assume that we want the boundary to be greater than or equal to ℓ_4. The axiom in ontology $\mathcal{O}_{Georgia}$ with label not greater than or equal to ℓ_4 is only o_2. If this axiom is relabeled to ℓ_4, then we obtain the meet of the MinA is ℓ_4. By the relabeling of axiom o_2, we obtain the infima of all MinAs as follows: ℓ_4, ℓ_5, ℓ_3, and ℓ_0. Therefore, the new boundary is ℓ_4.*

The axioms o_1 and o_4 in the MinA $\{o_1, o_2, o_4\}$ are considered not necessary to be relabeled. Their labels do not need to be changed in order to repair the boundary, because they are already greater than or equal to ℓ_4.

In other words, relabeling some essential axioms is enough to change the infimum of this MinA. To change the infimum of a MinA, we just need to change the label of axioms which are less than ℓ_g or incomparable to ℓ_g. The reason is that these axioms are the axioms which "responsible" to the infimum of the MinA. Therefore, relabeling them guarantees that the infimum of the MinA is greater than or equal to ℓ_g. The set of these axioms has been introduced previously in Definition 2.10. From Lemma 3.2 together with Lemma 2.12, it is easily proved that every IAS is a minimal change set for this condition. And for the case less than or equal to ℓ_g, we just need to relabel axioms which are not less than or equal to ℓ_g, which has been defined in Definition 2.10. And from the Lemma 3.3 and Lemma 2.12, RAS is the minimal change set for changing a boundary to less than or equal to ℓ_g.

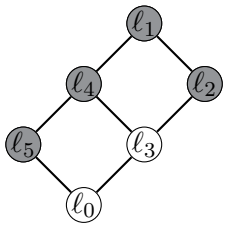


Figure 3.1: Labels satisfy condition not less than or equal to ℓ_3

3.2 Negative Conditions

In the previous section, we have shown how to deal with the positive conditions. The greater than or equal to case can be solved using the computation of an IAS as it is used in $\ell_g > \nu$ case in equality condition. And the less than or equal to case can be solved using the computation of an RAS as it is used in $\ell_g < \nu$ case in equality condition. Now we show how we can get a boundary which satisfies the conditions, i.e. the negation of less than or equal to and greater than or equal to. To obtain a change set for these conditions is not trivial at all. First, we show the naïve approach which is related to MinAs and diagnoses. Later, we explain how to get better solutions for these conditions.

For negative conditions we allow relabeling the change set to any other label which is in the given condition or criteria, instead of relabeling to the goal label ℓ_g. In the positive conditions, elements of the change set are relabeled to ℓ_g. This is not possible for the negative conditions, because labeling them to ℓ_g might give us a boundary that is equal to ℓ_g which we do not want. If the given condition is greater than or equal to ℓ, then the flexibility on relabeling the change set is to any label which is greater than or equal to ℓ.

3.2.1 Naïve Approach

When a knowledge engineer wants a boundary to be not less than or equal to ℓ_g, where ℓ_g is a goal label, he wants the boundary to be a label that is strictly greater than or incomparable to ℓ_g. For example, the grey nodes in Figure 3.1 shows which labels satisfy the condition not less than or equal to ℓ_3.

To obtain a boundary which is not less than or equal to ℓ_g, from Lemma 2.4 can be derived that a MinA of a given ontology $\mathcal{O}$ and a consequence c is a change set for this condition $\nu' \not\leq \ell_g$. If all axioms in a MinA are relabeled to a label $\not\leq \ell_g$, the infimum of the MinA is changed to that label. With the fact that all infima of the MinAs are less than or equal to the boundary and that the boundary is less than or equal to the goal value, the new boundary is not less than or equal to ℓ_g. The following lemma justifies this.

Lemma 3.5. *Let ℓ_g be a goal label, lab a labeling function, and $\mathcal{S}$ a MinA. $\mathcal{S}$ is a change set for changing a boundary $\nu \not\leq \ell_g$ by $\mathsf{lab}_{\mathcal{S},\ell}(\mathcal{O})$ where $\ell \in L$, $\ell \not\leq \ell_g$.*

Proof. We want to show that after relabeling a MinA $\mathcal{S}$ to a label $\ell \not\leq \ell_g$ we obtain a new boundary $\nu' \not\leq \ell_g$. Lemma 2.4 shows that if $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_n$ are all the MinAs for $\mathcal{O}$, c then $\bigoplus_{i=1}^{n} \lambda_{\mathcal{S}_i}$ is a boundary for $\mathcal{O}$, c. We assume w.l.o.g. that the previous computed boundary $\nu \leq \ell_g$, because if it is not less than or equal to ℓ_g, then we do not need to change anything. We have the following order

$$\forall i = 1, \ldots, n, \lambda_{\mathcal{S}_i} \leq \nu \leq \ell_g.$$

When we relabel any MinA $\mathcal{S}$ to a label $\ell \not\leq \ell_g$, there are two cases that we need to analyze.

ℓ is greater than ℓ_g If ℓ is greater than ℓ_g, then relabeling $\mathsf{lab}_{\mathcal{S},\ell}$ gives a $\lambda_{\mathcal{S}} > \ell_g$ which implies that the new boundary ν' (which is greater than or equal to $\lambda_{\mathcal{S}}$) is strictly greater than ℓ_g.

ℓ, ℓ_g are incomparable If ℓ is incomparable to ℓ_g, then relabeling $\mathsf{lab}_{\mathcal{S},\ell}$ gives a $\lambda_{\mathcal{S}}$ which is incomparable to ℓ_g. We know that a boundary is greater than or equal to $\lambda_{\mathcal{S}}$. If the previous boundary is equal to $\lambda_{\mathcal{S}}$, then ν' is incomparable to ℓ_g. If the previous boundary is greater than $\lambda_{\mathcal{S}}$, then ν' is either greater than or incomparable to ℓ_g.

Therefore, in both cases we obtain the new boundary ν' that is not less than or equal to ℓ_g. □

For obtaining a boundary that is not greater than or equal to ℓ_g, we can use the dual notion of MinA, i.e. diagnosis as shown on the following lemma.

Lemma 3.6. *Let ℓ_g be a goal label, lab a labeling function, and $\mathcal{S}$ a diagnosis. $\mathcal{S}$ is a change set for changing a boundary $\nu \not\geq \ell_g$ by $\mathsf{lab}_{\mathcal{S},\ell}(\mathcal{O})$ where $\ell \in L$, $\ell \not\geq \ell_g$.*

Proof. We want to show that after relabeling a diagnosis $\mathcal{S}$ to a label $\ell \not\geq \ell_g$ we obtain a new boundary $\nu' \not\geq \ell_g$. Lemma 2.4 shows that if $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_n$ are all the diagnoses for $\mathcal{O}$, c then $\bigotimes_{i=1}^{n} \mu_{\mathcal{S}_i}$ is a boundary for $\mathcal{O}$, c. We assume w.l.o.g. that the previous computed boundary $\nu \geq \ell_g$, because if it is not greater than or equal to ℓ_g, then we do not need to change anything. We have the following order

$$\forall i = 1, \ldots, n, \mu_{\mathcal{S}_i} \geq \nu \geq \ell_g.$$

When we relabel any diagnosis $\mathcal{S}$ to a label $\ell \not\geq \ell_g$, there are two cases that we need to prove.

ℓ is less than ℓ_g If ℓ is less than ℓ_g, then relabeling $\mathsf{lab}_{\mathcal{S},\ell}$ gives a $\mu_{\mathcal{S}} < \ell_g$ which implies that the boundary ν is smaller than or equal to $\mu_{\mathcal{S}}$. Therefore ν is strictly smaller than ℓ_g.

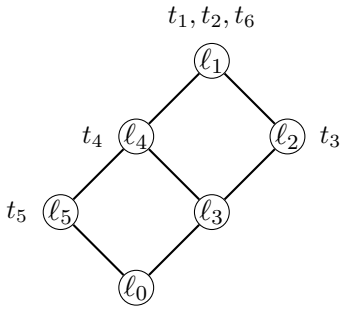


Figure 3.2: Labeled ontology w.r.t. the lattice.

ℓ, ℓ_g **are incomparable** If ℓ is incomparable to ℓ_g, then relabeling $\mathsf{lab}_{\mathcal{S},\ell}$ gives a $\mu_{\mathcal{S}}$ which is incomparable to ℓ_g. We know that the boundary $\nu \leq \mu_{\mathcal{S}}$. If $\nu = \mu_{\mathcal{S}}$, then ν is incomparable to ℓ_g. If $\nu < \mu_{\mathcal{S}}$, then ν is either smaller than or incomparable to ℓ_g.

Therefore, in both cases we obtain the boundary ν is a value $\not\geq \ell_g$. □

3.2.2 Label-Based Optimized Approach

We now discuss how to get the minimal change set for our current cases. The following example shows that in some cases a MinA is not the minimal change set for repairing the boundary to a value not less than or equal to ℓ_g.

Example 3.7. *Consider ontology $\mathcal{O}_{Georgia}$ and the boundary for $Georgia \sqsubseteq World$ is ℓ_3. We take a MinA $\{o_1, o_3, o_4\}$ from ontology $\mathcal{O}_{Georgia}$ as a change set for repairing the boundary to be $\not\geq \ell_2$. All axioms of this MinA which were previously labeled with $\{\ell_1, \ell_3, \ell_4\}$ is relabeled to ℓ_1. The new boundary is ℓ_1. This MinA is not minimal and also not the smallest change set. A minimal change set that we can get from this MinA is the subset $\{o_3\}$. When o_3 is relabeled to ℓ_1, the boundary becomes ℓ_4 which is not greater than or equal to ℓ_2. This result is obtained because for the MinA $\{o_1, o_3, o_4\}$, the meet of the labels of the axioms becomes ℓ_4 after relabeling o_3 to ℓ_1. Therefore, with the infimum of a MinA equal to ℓ_4, we obtain the boundary ℓ_4.*

As we know that a MinA or a diagnosis might not be the smallest change set, we need to find a minimal change set to avoid unnecessary changes to particular axioms. We can try to apply the same approach as mentioned in [KP10b] for these conditions. The idea is to ignore axioms of a MinA (diagnosis) that do not need to be relabeled, because their labels have followed the repairing condition. Therefore any axiom that has a label which follows the given condition, in this case $\not\leq$ or $\not\geq$ to ℓ_g are not contained in the change set.

In the changes, we will apply the target label to the change set. For the first case we choose ℓ_1 or the top element of the lattice as the target label when the condition is the new boundary not less than or equal to ℓ_g and ℓ_0 or the bottom element of the lattice when the condition is the new boundary not greater than or equal to ℓ_g.

Example 3.8. *Recall the boundary for the ontology $\mathcal{O}_{Georgia}$ and the consequence $Georgia \sqsubseteq World$ is ℓ_3 with respect to the labeling lattice $(L_d, \leq_d)$. Assume that the knowledge engineer wants to change the boundary to $\not\leq \ell_3$, then the question is what the smallest change set is for this condition. As it has been proven that a MinA is a change set in this case, let us take a MinA $\mathcal{S}$ from $\mathcal{O}$, c, say $\{o_1, o_3, o_4\}$. Changing all axioms in $\mathcal{S}$ to a label $\ell \not\leq \ell_3$, i.e. $\ell_1, \ell_2, \ell_4, \ell_5$ changes the boundary ℓ_1, ℓ_2, ℓ_4, or ℓ_5 respectively.*

We want to obtain the smallest change set from the same $\mathcal{S}$. When we consider the fixed axioms, i.e. axioms which do not need to be changed, the smallest change set is $\mathcal{S} \setminus \mathcal{O}_{\not\leq \ell_3}$, namely $\{o_3\}$. Then we apply labeling function $\mathsf{lab}_{\{o_3\},\ell}$ where $\ell \in \{\ell_1, \ell_2, \ell_4, \ell_5\}$, the boundary computed after the labeling $\mathcal{S}$ to all $\ell \not\leq \ell_3$, except to ℓ_2, fulfills the condition $\nu \not\leq \ell_3$. Relabeling the change set to label ℓ_2 does not change the meet of the MinA, because the infimum of labels of o_4 and o_3 is ℓ_3.

From Example 3.8, we know that labeling $\{o_3\}$ to ℓ_2 is not enough to repair the boundary to a value $\not\leq \ell_3$. We observe the effect of labeling o_3 to ℓ_2. The affected MinAs are $\{o_1, o_3, o_4\}, \{o_1, o_3, o_5\}$ which give $\lambda_{\{o_1,o_3,o_4\}} = \ell_3$, $\lambda_{\{o_1,o_3,o_5\}} = \ell_0$. Let us analyze MinA $\mathcal{S} = \{o_1, o_3, o_4\}$. The labels of these axioms before applying the labeling function are $\{\ell_1, \ell_3, \ell_4\}$. When we label o_3 to ℓ_2, the meet of $\{\ell_1, \ell_2, \ell_4\}$ is ℓ_3. For $\mathcal{S} = \{o_1, o_3, o_5\}$, we obtain $\lambda_{\mathcal{S}} = \ell_0$ after $\mathsf{lab}_{\{o_3\},\ell_2}$. Our idea of moving the $\lambda_{\mathcal{S}}$ to some greater or incomparable label is not applied here, because there is an axiom in the fixed portion which makes the infimum of the changed axiom $\leq \ell_g$, for example o_4 or o_5 together with the changed axiom o_3.

The following example shows a deeper problem in changing the boundary, in special case to repair the boundary to be not greater than or equal to a goal label.

Example 3.9. *Using the same lattice $(L_d, \leq_d)$, consider the ontology $\mathcal{T}$.*

$$
\begin{aligned}
t_1 : c_1 &\sqsubseteq c_2 \sqcap c_3, \\
t_2 : c_1 &\sqsubseteq c_5 \sqcap c_6, \\
t_3 : c_2 &\sqsubseteq c_7, \\
t_4 : c_3 &\sqsubseteq c_7, \\
t_5 : c_5 &\sqsubseteq c_7, \\
t_6 : c_6 &\sqsubseteq c_7.
\end{aligned}
$$

Each axiom is assigned to the label in the lattice as shown in Figure 3.2. Following the Lemma 2.4, we obtain the boundary for the consequence $c_1 \sqsubseteq c_7$ is ℓ_1.

Suppose the knowledge engineer wants to change the boundary to a value $\not\geq \ell_1$. From Lemma 3.6, we know that any diagnosis is a change set. We take the diagnosis $\{t_2, t_3, t_4\}$. The sub-ontology $\mathcal{T}_{\not\geq \ell_1}$ is considered to be a set of unchanged axioms. From the diagnosis, we need only to change $\{t_2\}$. Unfortunately, it is not enough to get a boundary $\not\geq \ell_1$ by changing $\{t_2\}$ to any label, because when we relabel the set $\{t_2\}$ to either $\ell_2, \ell_3, \ell_4, \ell_5, \ell_5$, the join of the diagnis is still ℓ_1.

The minimal change sets from this diagnosis are $\{t_2, t_3\}$ and $\{t_2, t_4\}$. If the labels of one of these sets are changed to ℓ_0, then the boundary with respect to the change is not greater than or equal to ℓ_1. And if we choose ℓ_5 as the target. If the knowledge engineer chooses to change a change set to ℓ_5, then only the change set $\{t_2, t_3\}$ will satisfy the criteria given by the engineer.

There are two main matters which are shown in the example. The first one is to obtain a set of axioms which need to be changed. This change set cannot be obtained trivially by considering the fixed part of an ontology and taking the non-fixed part of a MinA or a diagnosis. The second problem is to select the right change set w.r.t. the target label. Both problems are related to one to each other. Therefore, to have better understanding, we first take the top or bottom element as the target label, because they are the identity elements in a lattice.

In order to change the boundary to a value $\not\geq \ell_g$, we need to have the supremum of a diagnosis which is not greater than or equal to ℓ_g. This means there should not be an subset of the diagnosis where the supremum of the elements is greater than or equal to ℓ_g. This can be ensured by checking if there exists an axiom with this label or if there is a minimal subset of the diagnosis which supremum is greater than or equal to ℓ_g. If an axiom with label greater than or equal to ℓ_g exists, then this axiom is included in the change set. If a subset with the supremum greater than or equal to ℓ_g exists, then one of axioms in this subset is included in the change set. In this way, we ensure that the supremum of the rest axioms is not greater than or equal to ℓ_g.

Recall the previous Example 3.9, the set $\{t_2\}$ is not a change set for obtaining the boundary to be not greater than or equal to ℓ_1. The diagnosis that we took is $\{t_2, t_3, t_4\}$. The label of t_2 is ℓ_1. Thus, t_2 is added in to the change set. Although the labels of t_3 and t_4 are ℓ_2 and ℓ_4, the supremum of them is ℓ_1. We have to avoid the possibility of a supremum of any subset of the diagnosis that is greater than or equal to ℓ_1. Therefore, we add either t_3 or t_4 in to the change set which makes the change set is either $\{t_2, t_3\}$ or $\{t_2, t_4\}$.

Based on this explanation, our approach to solve this problem is that first we compute the set of labels of axioms in the change set, then we take only axioms which are labeled with an element in that set. The set of labels is called solution label set.

Definition 3.10 (NGEQ Solution label set)**.** *Let D be a diagnosis for a given ontology $\mathcal{O}$ and consequence c, D_{lab} be the set of labels of axioms in D, and*

ℓ_g a goal label. A NGEQ solution label set *is a subset $S_{lab} \subseteq D_{lab}$ where $\bigoplus_{\ell \in (D_{lab} \setminus S_{lab})} \ell \not\geq \ell_g$. It is called a* minimal NGEQ solution label set *if $\forall S' \subset S_{lab}$, $\bigoplus_{\ell \in (D_{lab} \setminus S')} \ell \geq \ell_g$.*

Given a solution label set, we can obtain a subset of the set of axioms which are labeled with elements from the solution label set. This subset contains only axioms which are contained in a respective diagnosis.

Definition 3.11 (NGEQ Solution set)**.** *Let $\mathcal{O}$ be an ontology, c a consequence, ℓ_g a goal label, D be a diagnosis for $\mathcal{O}$ and c, D_{lab} be a set of labels of axioms in D, $S_{lab} \subseteq D_{lab}$ a solution label set for new boundary $\nu' \not\geq \ell_g$, and $\mathcal{O}_{fixed} = \mathcal{O}_{(D_{lab} \setminus S_{lab})}$. A* NGEQ solution set *is a subset $S \subseteq \mathcal{O}_{S_{lab}}$ such that $\mathcal{O} \setminus (S \cup \mathcal{O}_{fixed}) \not\models c$. It is a* minimal NGEQ solution set *if $\forall S' \subset S$, $\mathcal{O} \setminus (S' \cup \mathcal{O}_{fixed}) \models c$.*

The following lemma justifies that a NGEQ solution set is a change set for condition not greater than or equal to ℓ_g.

Lemma 3.12. *Let $\mathcal{C}$ be a minimal change set for changing the boundary to $\nu \not\geq \ell_g$ for ontology $\mathcal{O}$ and a consequence c, $\mathcal{C}_{lab}$ be a set of labels of axioms in $\mathcal{C}$, ℓ_t a target label, then there exists a NGEQ solution set $\mathcal{S}$ such that $\mathcal{S} = \mathcal{C}$.*

Proof. By Lemma 2.4, we know that the boundary is less than or equal to $\mu_{\mathcal{S}}$ of any diagnosis of $\mathcal{O}$ and c. And by definition of the change set, under the labeling function $\mathsf{lab}_{\mathcal{C},\ell_t}$ the boundary is changed to a value $\not\geq \ell_g$. Therefore, under $\mathsf{lab}_{\mathcal{C},\ell_0}$, there exist a diagnosis D where $(\bigoplus_{\ell \in D} \ell) \not\geq \ell_g$. That means the new supremum (after we relabel it to ℓ_0) is not greater than or equal to ℓ_g. If we take any subset $\mathcal{C}' \subset \mathcal{C}$, then $\mu_D \geq \ell_g$ under $\mathsf{lab}_{\mathcal{C}',\ell_t}$. As the consequence, every axiom in $\mathcal{C}$ is also in D. Therefore, it is true that $\mathcal{C} \subseteq D$.

It is trivial that $\mathcal{C}_{lab} \subseteq D_{lab}$ where D_{lab} is the set of labels of axioms in D. $\bigoplus_{\ell \in (D_{lab} \setminus \mathcal{C}_{lab})} \ell \not\geq \ell_g$ is true because of the fact that $\mathcal{C}$ is a minimal change set and the new boundary ν' is not greater than or equal to ℓ_g under the labeling function $\mathsf{lab}_{\mathcal{C},\ell_t}$. Therefore, $\mathcal{C}_{lab}$ is a solution label set.

We know that from $\mathcal{C}$ there exists a solution label set, i.e. $\mathcal{C}_{lab}$. As a consequence, $\mathcal{C}$ is a solution set w.r.t $\mathcal{C}_{lab}$. Since $\mathcal{C}_{lab}$ is a solution label set, there exists a solution set, i.e. $\mathcal{C}$. □

For the case where the boundary is required to be not less than or equal to a goal label, a change set can be obtained in an analogous way. In this case, a minimal axiom set is a change set as it has been shown in Lemma 3.5. We have to change a MinA in such way its λ is not less than or equal to a goal label. That means some axioms in a MinA are responsible for its infimum to be less than or equal to the goal label. Similar to the previous case, we can find these axioms by looking at theirs labels first, and then filter the axioms from these labels.

Definition 3.13 (NLEQ Solution label set and NLEQ Solution set)**.** *Let M_{lab} be a set of labels of axioms in a minimal axiom set for a given ontology $\mathcal{O}$, consequence c, ℓ_g a goal label. A* NLEQ solution label set *is a subset $S_{lab} \subseteq M_{lab}$ where $\bigotimes_{\ell \in (M_{lab} \setminus S_{lab})} \ell \not\leq \ell_g$. It is called a* minimal NLEQ solution label set *if $\forall S' \subset S_{lab}$, $\bigotimes_{\ell \in (M_{lab} \setminus S')} \ell \leq \ell_g$.*

A NLEQ solution set *is a subset $S \subseteq \mathcal{O}_{S_{lab}}$ such that $(S \cup \mathcal{O}_{fixed}) \models c$, where $\mathcal{O}_{fixed} = \mathcal{O}_{(D_{lab} \setminus S_{lab})}$. It is a* minimal NLEQ solution set *if $\forall S' \subset S$, $(S' \cup \mathcal{O}_{fixed}) \not\models c$.*

The following lemma is the dual of Lemma 3.12 for showing that a minimal change set is a minimal NLEQ solution set for changing the boundary to be not less than or equal to a goal label. The proof can be obtained in analogous way as in the proof of Lemma 3.12.

Lemma 3.14. *Let $\mathcal{C}$ be a minimal change set in order to change the boundary $\nu \not\leq \ell_g$ for ontology $\mathcal{O}$ and a consequence c, $\mathcal{C}_{lab}$ be a set of labels of axioms in $\mathcal{C}$, then there exists a NLEQ solution set $\mathcal{S}$ such that $\mathcal{S} = \mathcal{C}$.*

The second case of the problems is how to select the right change set w.r.t. the target label. That means we want to change the axioms in a change set as close as possible to their original labels. Previously, we fixed the target label to be the top or bottom element of the lattice which is the identity element for each of join and meet operations. This was done to simplify the problem. Unfortunately, relabeling to the top or bottom element often changes more the level of restriction of the axioms in the change set which we can reduce by relabeling to a label which is greater than the bottom element or less than the top element. When an axiom is changed to the bottom element, the axiom is set to be as private as possible or if an axiom is relabeled to the top element, it is set to be accessible by every user.

The computation of a change set has always been respecting the target label. When a label which is not the top or bottom element is chosen as the target label, the computation is a special case because the top or bottom element is an identity element in the lattice. Thus, if we computed a change set with top (bottom) element as the target label, the set could not satisfy the property of a change set. As we have seen in the previous example that the set $\{t_2, t_4\}$ is not a change set if the target label is ℓ_5 for a given change condition boundary not greater than or equal to ℓ_1, i.e. after labeling $\{t_2, t_4\}$ to ℓ_5, the supremum of the diagnosis $\{t_2, t_3, t_4\}$ is $\ell_5 \oplus \ell_2 \oplus \ell_5 = \ell_1$.

Taking the target label into the computation, we modify the definitions of solution label set. The supremum or infimum of the solution label set together with the target label should not be greater or less than the goal label.

Definition 3.15 (NGEQ and NLEQ solution label set revised)**.** *Let $\mathcal{O}$ be an ontology, c a consequence, D_{lab} be a set of labels of axioms in a diagnosis for*

Algorithm 6: HST algorithm for computing diagnosis label

Procedure hst-Diagnosis-Label($\mathcal{O}, c, \ell_g, \ell_t$)
Input: $\mathcal{O}$: ontology; c: consequence; ℓ_g: goal label; ℓ_t: target label
Output: a solution set

1: **Global**: **C, H, Sol** := $\emptyset$
2: hst-diag-lab($\mathcal{O}, c, \ell_g, \ell_t, \emptyset$)
3: **return** min(Sol) (the min function returns the smallest set in **Sol**)

Procedure hst-diag-lab($\mathcal{O}, c, \ell_g, \ell_t, H$)
Input: $\mathcal{O}$: ontology; c: consequence; ℓ_g: goal label; ℓ_t: target label; H: set of axioms $\subseteq \mathcal{O}$
Side effects: modifications to **C**, **H** and **Sol**

4: **if** there exists some $H' \in \mathbf{H}$ such that $H' \subseteq H$ **or** H' contains a prefix-path P with $P = H$ **then**
5: **return** (early path termination)
6: D := compute-diagnosis-label($\mathcal{O}, c, H$)
7: **if** $H \not\models c$ **then**
8: $\mathbf{C} := \mathbf{C} \cup \{D\}$
9: Sol := hst-Solution-Label-of-Diagnosis(D, ℓ_g, ℓ_t, H)
10: $\mathbf{Sol} := \mathbf{Sol} \cup Sol$
11: $\mathcal{A} := \{a \in sol \mid sol \in Sol\}$
12: **foreach** *axiom* $a \in \mathcal{A}$ **do**
13: hst-diag-lab($\mathcal{O}, c, \ell_g, \ell_t, H \cup \{a\}$)
14: **else**
15: $\mathbf{H} := \mathbf{H} \cup \{H\}$

Procedure compute-diagnosis-label($\mathcal{O}, c, H$)
Input: $\mathcal{O}$: ontology; c: consequence; H: set of axioms $\subseteq \mathcal{O}$
Output: a set of labels of a diagnosis

16: **if** $H \models c$ **then**
17: **return** *no diagnosis label*
18: $\mathcal{S} := \mathcal{O} \setminus H$
19: $D := \emptyset$
20: $L := L_{\mathsf{lab}}(\mathcal{S})$
21: **foreach** label $k \in L$ **do**
22: **if** $\mathcal{O} \setminus \{\mathcal{S}_{\neq k}\} \not\models c$ **then**
23: $\mathcal{S} := \mathcal{S}_{\neq k}$
24: **else**
25: $D := D \cup \{k\}$
26: **return** D

$\mathcal{O}$ and c, M_{lab} be a set of labels of axioms in a MinA for $\mathcal{O}$ and c, ℓ_g a goal label, ℓ_t a target label. A NGEQ solution label set *is a subset $S_{lab} \subseteq D_{lab}$ where $\bigoplus_{\ell \in (D_{lab} \setminus S_{lab}) \cup \{\ell_t\}} \ell \not\geq \ell_g$. It is called a* minimal NGEQ solution label set *if $\forall S' \subset S_{lab}$, $\bigoplus_{\ell \in (D_{lab} \setminus S') \cup \{\ell_t\}} \ell \geq \ell_g$.*

A NLEQ solution label set *is a subset $S_{lab} \subseteq M_{lab}$ where $\bigotimes_{\ell \in (M_{lab} \setminus S_{lab}) \cup \{\ell_t\}} \ell \not\leq \ell_g$. It is called a* minimal NLEQ solution label set *if $\forall S' \subset S_{lab}$, $\bigotimes_{\ell \in (M_{lab} \setminus S') \cup \{\ell_t\}} \ell \leq \ell_g$.*

Using the revised definitions of NGEQ and NLEQ solution label set, we can compute a change set with considering the target label. The following section describes how to obtain the smallest change set based on the lemmas that we

Algorithm 7: HST algorithm for computing solution label set

Procedure hst-Solution-Label-of-Diagnosis($D, \ell_g, \ell_t, \mathcal{O}_{removed}$)
Input: D: set of labels of diagnosis; ℓ_g: goal label; ℓ_t: target label; $\mathcal{O}_{removed}$: set of removed axioms
Output: set of solution sets

1: **Global**: **C, H, Sol** := $\emptyset$
2: hst-sol-lab($D, \ell_g, \ell_t, \emptyset, \mathcal{O}_{removed}$)
3: **return Sol**

Procedure hst-sol-lab($D, \ell_g, \ell_t, H, \mathcal{O}_{removed}$)
Input: D: set of labels of diagnosis; ℓ_g: goal label; ℓ_t: target label; H: set of removed labels $\subseteq D$; $\mathcal{O}_{removed}$: set of removed axioms
Side effects: modifications to **C**, **H** and **Sol**

4: **if** there exists some $H' \in \mathbf{H}$ such that $H' \subseteq H$ **or** H' contains a prefix-path P with $P = H$ **then**
5: **return** (early path termination)
6: **if** there exists some $C' \in \mathbf{C}$ such that $C' \cap H = \emptyset$ **then**
7: $S := C'$ (solution label set reuse)
8: **else**
9: $S :=$ compute-solution-label(D, ℓ_g, ℓ_t, H)
10: **if** $\bigoplus_{\ell \in \{(D \setminus S) \cup \{\ell_t\}\}} \ell \not\geq \ell_g$ **then**
11: $\mathbf{C} := \mathbf{C} \cup \{S\}$
12: $Sol :=$ hst-Solution-of-Diagnosis($\mathcal{O}, \mathcal{O}_S \setminus \mathcal{O}_{removed}, \mathcal{O}_{(D \setminus S)}, c$)
13: **Sol** := **Sol** $\cup \{Sol\}$
14: **foreach** label $\ell \in S$ **do**
15: hst-sol-lab($D, \ell_g, \ell_t, H \cup \{\ell\}, \mathcal{O}_{removed}$)
16: **else**
17: **H** := **H** $\cup \{H\}$

Procedure compute-solution-label($D, \ell_g, \ell_t, D_{removed}$)
Input: D: set of labels of diagnosis; ℓ_g: goal label; ℓ_t: target label; $D_{removed}$: set of removed labels $\subseteq D$
Output: a solution label set

18: **if** $\bigoplus_{\ell \in (D_{removed} \cup \{\ell_t\})} \ell \geq \ell_g$ **then**
19: **return** *no solution label set*
20: $T := D \setminus D_{removed}$
21: $S := \emptyset$
22: **foreach** label $\ell \in T$ **do**
23: **if** $\bigoplus_{k \in (D \setminus \{T - \ell\} \cup \{\ell_t\})} k \not\geq \ell_g$ **then**
24: $T := T \setminus \{\ell\}$
25: **else**
26: $S := S \cup \{\ell\}$
27: **return** S

have mentioned.

3.3 Computing the Smallest Change Set

If our target is to obtain a change set for a given condition, the simplest way is to take the whole ontology. One step further is to take a minimal axiom set or a diagnosis as the change set. We have four conditions depending on the label

Algorithm 8: HST algorithm for computing solution set

Procedure hst-Solution-of-Diagnosis($\mathcal{O}, \mathcal{O}_{slab}, \mathcal{O}_{fixed}, c$)
Input: $\mathcal{O}$: ontology; $\mathcal{O}_{slab}$: ontology w.r.t a solution label set; $\mathcal{O}_{fixed}$: ontology w.r.t. diagnosis label set minus solution label set; c: a consequence of $\mathcal{O}$
Output: set of solution sets

1: **Global**: **C, H** := $\emptyset$
2: hst-sol($\mathcal{O}, \mathcal{O}_{slab}, \mathcal{O}_{fixed}, c, \emptyset$)
3: **return C**

Procedure hst-sol($\mathcal{O}, \mathcal{O}_{slab}, \mathcal{O}_{fixed}, c, H$)
Input: $\mathcal{O}$: ontology; $\mathcal{O}_{slab}$: ontology w.r.t a solution label set; $\mathcal{O}_{fixed}$: ontology w.r.t. diagnosis label set minus solution label set; c: a consequence of $\mathcal{O}$; H: set of removed axioms
Side effects: modifications to **C** and **H**

4: **if** there exists some $H' \in \mathbf{H}$ such that $H' \subseteq H$ **or** H' contains a prefix-path P with $P = H$ **then**
5: **return** (early path termination)
6: **if** there exists some $C' \in \mathbf{C}$ such that $C' \cap H = \emptyset$ **then**
7: $S := C'$ (solution set reuse)
8: **else**
9: S := compute-solution($\mathcal{O}, \mathcal{O}_{slab}, \mathcal{O}_{fixed}, c$)
10: **if** $\mathcal{O} \setminus \{\mathcal{O}_{slab} \cup \mathcal{O}_{fixed}\} \not\models c$ **then**
11: **C** := **C** $\cup\{S\}$
12: **foreach** axiom $o \in S$ **do**
13: hst-sol($\mathcal{O}, \mathcal{O}_{slab} \setminus \{\ell\}, \mathcal{O}_{fixed}, c, H \cup \{\ell\}$)
14: **else**
15: **H** := **H** $\cup\{H\}$

Procedure compute-solution($\mathcal{O}, \mathcal{O}_{slab}, \mathcal{O}_{fixed}, c$)
Input: $\mathcal{O}$: ontology; $\mathcal{O}_{slab}$: ontology w.r.t a solution label set; $\mathcal{O}_{fixed}$: ontology w.r.t. diagnosis label set minus solution label set; c: a consequence of $\mathcal{O}$
Output: a solution set

16: **if** $\mathcal{O} \setminus \{\mathcal{O}_{slab} \cup \mathcal{O}_{fixed}\} \models c$ **then**
17: **return** *no solution set*
18: $\mathcal{T} := \mathcal{O}_{slab}$
19: $\mathcal{S} := \emptyset$
20: **foreach** axiom $o \in T$ **do**
21: **if** $\mathcal{O} \setminus \{\mathcal{T} \cup \mathcal{O}_{fixed} \setminus \{o\}\} \not\models c$ **then**
22: $\mathcal{T} := \mathcal{T} \setminus \{o\}$
23: **else**
24: $S := S \cup \{o\}$
25: **return** S

where the boundary is conditioned to be. On previous work [KP10b, KP10a], hitting set tree (HST) algorithm was used to compute inserted axioms sets (IAS) and removed axioms sets (RAS) in order to change the boundary depending the given goal label. As we have shown previously, to change the boundary to be greater or equal to a goal label or to be smaller or equal to a goal label, an IAS or RAS is a minimal change set. Using the HST algorithm introduced in [KP10b], (partial) IAS and RAS can be computed. Once the termination condition, i.e. when the algorithm can not find any smaller IAS or RAS, is fulfilled, the smallest

change set is obtained.

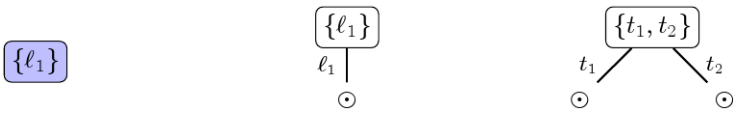
Our approach for the condition not greater than or equal to a goal label or not less than or equal to a goal label is directly computing the set of labels in the HST algorithms, namely Algorithm 6 and 7. This minimizes the consumed time to get the smallest change set. The procedure hst-Diagnosis-Label computes the set of labels of diagnosis. Then, to get a minimal change set, the procedure compute-solution-label-of-diagnosis finds in which label the axioms are.

For the condition not greater than or equal to a goal label or not less than or equal to a goal label, it has been shown in Lemma 3.5 and 3.6 that a diagnosis (a MinA) is a change set and that we can obtain a minimal change set from it. The computation of the minimal change set for these particular conditions can be done by first computing a diagnosis for the ontology and the consequence. A solution label set of the diagnosis can be computed by ensuring that, without any axiom which label is in the solution label set, the supremum of labels of the rest of the axioms in the diagnosis is not greater than or equal to the goal label. It is also necessary to take the target label into account. We have to check if the supremum of labels of the rest of the axioms in the diagnosis together with the target label is not greater than or equal to the goal label.

In Algorithm 6, hst-Diagnosis-Label computes all sets labels in diagnosis, that is, the set of labels of all axioms in the diagnosis. Given an ontology $\mathcal{O}$ and a consequence c, the procedure compute-diagnosis-label extracts a set of labels of diagnosis. Then, the result is sent to procedure hst-Solution-Label-of-Diagnosis shown in Algorithm 7. The parameters required for this procedure are the set of labels of a diagnosis, a goal label, a target label, and a set of axioms which is removed from the original ontology. There are some minimal solution sets in a solution label set, so we ensure that every computed solution set on the same branch is different one to another. The compute-solution-label-of-diagnosis is used to obtain one solution label set in the diagnosis. The Algorithm 8 computes all solution sets in this diagnosis with a reduced search space, namely a set of axioms whose labels are in the previously computed solution label set $\mathcal{O}_{S_{lab}}$. Taking $\mathcal{O}$, $\mathcal{O}_{S_{lab}}$, $\mathcal{O}_{fixed} = \mathcal{O}_{D_{lab} \setminus S_{lab}}$, and the consequence c, the procedure compute-solution-of-diagnosis computes a single solution set, which is a minimal change set. Then, the procedure hst-Diagnosis-Label will recursively call itself to expand the hitting set tree where each branch will have a node which computed by removing one axiom from the computed solution set. This means that we will get new solution sets from a new branch where one axiom is not in these solution sets.

For each of negative conditions, there are three nested hitting set tree algorithms. In the case to change a boundary to be not greater than or equal to a goal label, the first HST algorithm is used to compute sets of labels in diagnosis. Each set computed is then used as a parameter to the second HST algorithm,

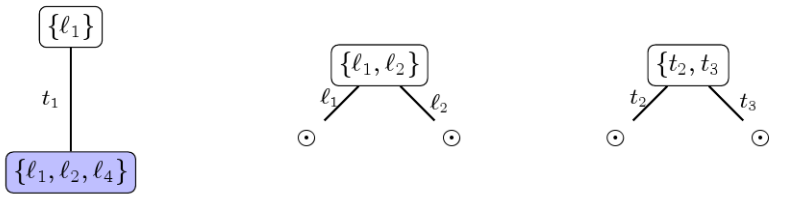
which computes sets of solution label set of the diagnosis. Later from each set of the solution label set, the third HST algorithm computes sets of solution set. The following example shows how the algorithm works.



(a)D_{lab} tree (b)S_{lab} tree (c)S tree

Figure 3.3: Hitting Set Trees from Algorithms 6, 7, and 8.

Example 3.16. *We reuse the ontology T which was used in Example 3.9 to show the algorithms work to get the smallest change set for changing the boundary to a value $\not\geq \ell_2$ with target label $\ell_t = \ell_3$. The process starts with procedure* hst-Diagnosis-Label$(\mathcal{T}, c_1 \sqsubseteq c_7, \ell_2, \ell_3)$. *It has three global variables* **C**, *which contains all sets of diagnosis labels,* **H**, *which contains sets of list of axioms from root to a single branch which has been terminated, and* **Sol**, *which contains all the solution sets computed in the process. They are initialized empty.*

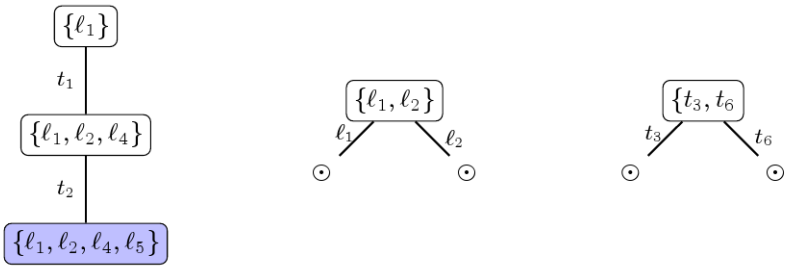


(a)D_{lab} tree (b)S_{lab} tree (c)S tree

Figure 3.4: The computation solution sets from node $\{\ell_1, \ell_2, \ell_4\}$.

The procedure hst-diag-lab *calls* compute-diagnosis-label$(\mathcal{T}, c_1 \sqsubseteq c_7, \emptyset)$ *which returns a set of labels of a diagnosis, say $D = \{\ell_1\}$. The root node is then labeled with $\{\ell_1\}$. The procedure* compute-diagnosis-label *computes the set D by trying to remove each label and then checking if the ontology without any axioms of this label still entails the consequence or not. The variable* **C** *is updated to $\{\{\ell_1\}\}$. The set $\{\ell_1\}$ is passed to procedure* hst-Solution-Label-of-Diagnosis$(\{\ell_1\}, \ell_2, \ell_3, \emptyset, \emptyset)$ *shown on line 9. In Line 9 of* hst-sol-lab, *variable S contains $\{\ell_1\}$ as the result of the procedure* compute-solution-label$(\{\ell_1\}, \ell_2, \ell_3, \emptyset)$. *This procedure computes the solution label set by removing each label from previous diagnosis label set and checking if the supremum of the rest labels is greater than or equal to ℓ_g, then the removed label is added to the variable S. After obtaining a solution label set,*

hst-sol-lab *calls Algorithm 8, with parameters* $(\mathcal{T}, \{t_1, t_2, t_6\}, \emptyset, c_1 \sqsubseteq c_7)$. *The procedure* hst-sol *computes all the solution sets which are subset of* $\{t_1, t_2, t_6\}$. *There is only one solution set, namely* $\{t_1, t_2\}$. *The variable* **Sol** *is updated to* $\{\{t_1, t_2\}\}$. *Figure 3.3 shows the data structures at this point.*



(a)D_{lab} tree (b)S_{lab} tree (c)S tree

Figure 3.5: The computation solution sets from node $\{\ell_1, \ell_2, \ell_4, \ell_5\}$.

After obtaining the solution set $\{t_1, t_2\}$, *the process returns to procedure* hst-Diagnosis-Label. *The procedure expands the hst tree for each elements in the solution set. It calls itself with parameters* $(\mathcal{T}, c_1 \sqsubseteq c_7, \ell_2, \ell_3, \{t_1\})$ *for edge* t_1 *and* $(\mathcal{T}, c_1 \sqsubseteq c_7, \ell_2, \ell_3, \{t_2\})$ *for edge* t_2. *First, we trace the edge* t_1. *The procedure obtains a diagnosis label set, say* $\{\ell_1, \ell_2, \ell_4\}$ *which labels the current node. This set is sent as the parameter of procedure* hst-Solution-Label-of-Diagnosis$(\{\ell_1, \ell_2, \ell_4\}, \ell_g, \ell_t, \{t_1\})$. *This time the procedure* compute-solution-label *returns* $\{\ell_1, \ell_2\}$. *At this point, it calls* hst-Solution-of-Diagnosis$(\mathcal{T}, \{t_2, t_3, t_6\}, \{t_4\}, c_1 \sqsubseteq c_7)$ *which returns* $\{t_2, t_3\}$. *Figure 3.4 shows the current stage.*

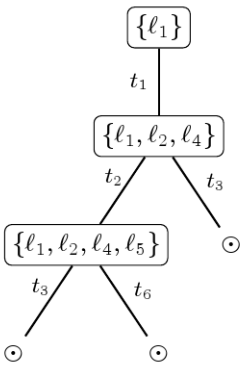
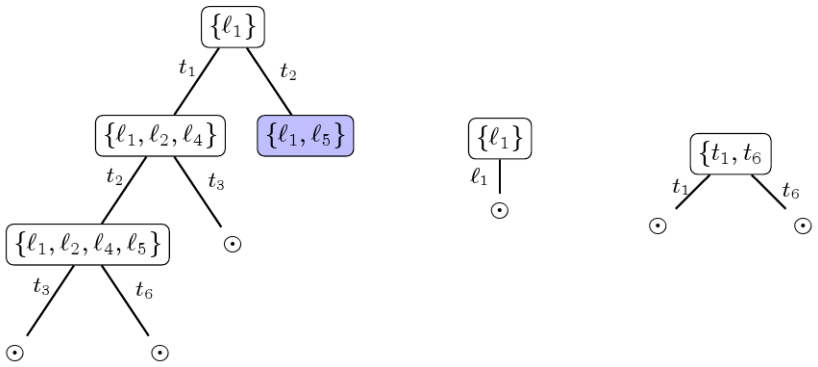


Figure 3.6: HST tree of diagnoses label.

Like before, Algorithm 6 expands the node $\{\ell_1, \ell_2, \ell_4\}$. *Each edge is labeled with one of the elements in* $\{t_2, t_3\}$. *On the edge* t_2 *which in the path* $\{t_1, t_2\}$, *the process obtains a new diagnosis label set, e.g.* $\{\ell_1, \ell_2, \ell_4, \ell_5\}$, *and at the end we*

obtain a new solution set, say $\{t_3, t_6\}$. *The expansions of the node* $\{\ell_1, \ell_2, \ell_4, \ell_5\}$ *are terminated normally. The paths* $\{t_1, t_2, t_3\}$, $\{t_1, t_3\}$, $\{t_1, t_2, t_6\}$ *are stored in* **H**. *Figure 3.5 shows the current stage.*



(a)D_{lab} tree (b)S_{lab} tree (c)S tree

Figure 3.7: The computation solution sets from node $\{\ell_1, \ell_5\}$.

At this point, **Sol** *contains three minimal change sets. They are* $\{\{t_1, t_2\}, \{t_2, t_3\}, \{t_3, t_6\}\}$ *as depicted in Figure 3.6.*

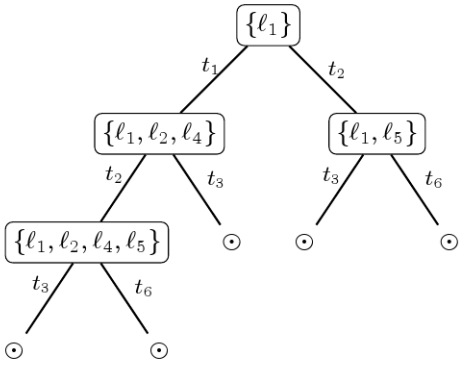


Figure 3.8: The complete HST tree of diagnoses label.

Back to the root node $\{\ell_1\}$, *now we trace the expansion with edge* t_2. *Procedure* compute-diagnosis-label$(\mathcal{T}, c_1 \sqsubseteq c_7, \{t_2\}$ *returns* $\{\ell_1, \ell_5\}$. *We obtain one solution set from this diagnosis label set, e.g.* $\{t_1, t_6\}$. *The expansions of this node are also terminated. The path* $\{t_2, t_1\}$ *is the early path terminated, since we have a path which has a prefix path equals to it, namely* $\{t_1, t_2, t_3\}$. *And the path* $\{t_2, t_6\}$ *terminates normally, because* $\{t_2, t_6\}$ *entails the consequence. Figure 3.7 shows the current stage.*

After tracing all the paths, we have four minimal change sets $\{\{t_1, t_2\}, \{t_2, t_3\}, \{t_3, t_6\}, \{t_1, t_6\}\}$. *They have the same number of axioms, so any of them is the smallest change set we can have. Figure 3.8 shows the final stage.*

The HST algorithm to compute sets of labels in diagnoses (MinAs) is implemented without the reuse node optimization. This optimization is not applicable here since the element used to label the node and the element used to label the edge are not the same, i.e. set of labels and set of axioms. Therefore, the condition for the optimization, $H(n') \cap S = \emptyset$ for node n', is simply not applicable. The other two HST algorithms for computing sets of solution label set and sets of solution set are optimized with early path termination similar to what has been used in [BKP09, KPHS07].

The HST algorithm that computes the diagnosis (MinA) label sets is implemented without the optimization of getting a particular set up to the m size, where m is a variable that contains the cardinality of the smallest change set computed on one stage. For example, we have two diagnosis label sets A and B. The diagnosis label set A has 2 labels. The relevant minimal change set is a set with 5 axioms. On the other hand, the diagnosis label B has 3 labels with a size-3-axioms minimal change set. Assume that on Algorithm 6 has this optimization. On the first computation of the diagnosis labels set, we obtain diagnosis A. The variable m is then updated to 2. That means on the next search, we want only a diagnosis labels set with cardinality up to 2. Therefore, we could never obtain the the minimal change set with size 3 axioms.

The same case applies to the HST algorithm that computes the solution label sets. Thus, this optimization is not applicable on both of the algorithms. In order to have a generality on the structure of the HST algorithms, we decided not to apply this optimization to the HST algortihm that computes the solution sets.

Another way to get the smallest change set without using the three-nested-HST algorithms is using one HST algorithm with the auxiliary black-box approach procedure. The idea is as follows. The auxiliary procedure checks on each sub-ontology if it is a change set. That means it takes one sub-ontology, then relabels it, and sees if the boundary is changed to the right condition. Although it seems that it will work, this way is very expensive one. If an ontology has a million of axioms, then to find the smallest change set for one consequence can take a long time and it requires a powerful tool to do the computation.

Chapter 4

Empirical Evaluation

In this chapter, we present the empirical evaluation of the implementation of the algorithms to repair a boundary of a consequence under different conditions. The implementation is built using Java 1.6 with Pellet 2.0 as the reasoner and OWL API trunk reversion 1150 as the JAVA API for manupulating the OWL Ontologies.

We have built one implementation for the test purpose and one implementation for user interface purpose. Each of the implementations requires an ontology or a set of modularized ontologies whose axioms are labeled by elements of the given lattice, an ontology that represents a lattice, a collection of consequences, and a collection of precomputed boundary of each consequence as the inputs. For the test purpose implementation, the output is a file .txt which contains the data obtained from the computation. For each computation of a given consequence and a condition, ee collect the time, number of minimal change set computed, the cardinality of the smallest change set, the interrupted fact, and, nevertheless, the smallest change set or the minimal change set which might not be the smallest one because of the interruption. For the user interface purpose, we extend the previous work, i.e. LBLR: A Lattice Based Labelled Reasoner.* Once the boundaries of the given consequences are computed, the user can get a change set for changing it to the conditions available.

The following sections describe the setup and the test case and plan.

4.1 Setup

The evaluation is done in a desktop computer. It has the following specification:

*LBLR is used to compute the boundary with an option to use one of three reasoners. See http://lat.inf.tu-dresden.de/systems/LBLR/

- CPU: Intel(R) Core(TM)2 Duo CPU E8500 3.16GHz
- RAM: 2.00GB
- Storage: 135.3 GB
- Operating system: Ubuntu release 8.10
- Java: Java 1.6
- Reasoner: Pellet 2.0.0 rc7
- OWL API trunk revision 1150

4.2 Evaluation Test Case and Plan

We used the same ontologies used in [BKP09, KP10a, KP10b]. The ontologies are called O^{SNOMED} and O^{FUNCT}. O^{SNOMED} is the Systematized nomenclature of Medicine, Clinical Terms (O^{SNOMED})[sno] which is a comprehensive medical and clinical ontology built using the Description Logic (DL) $\mathcal{EL}^+$. It contains 379,691 concept names, 62 object property names, and 379,704 axioms. It entails more than five million subsumptions. In this evaluation, we used a sampled set of 807 subsumptions. O^{FUNCT} is an OWL DL ontology for functional description of mechanical engineering solutions presented in [GKL09]. It contains 115 concept names, 47 object property names, 16 data property names, 545 individual names, 3,176 axioms. It is built using $\mathcal{SHOIN}(\mathbf{D})$, and entails 12 subsumption and 704 instance relationships (class assertions). We used a sample set which contains 4 subsumption and 43 instance relationship.

Both ontologies were labeled randomly using six elements $\{\ell_0, \ldots, \ell_5\}$. We limit the number of computed change sets to 10. For conditions greater than or equal to and less than or equal to, we use the algorithms introduced in [KP10a]. And once we find a change set with size 1, the computation is stopped.

We use five lattices. The first three lattices have six elements $\{\ell_0, \ldots, \ell_5\}$. The lattice $(L_d, \leq_d)$ is shown in Figure 2.1. The lattice $(L_l, \leq_l)$ is a linear order lattice where $\leq_l := \{(\ell_n, \ell_{n+1}) \mid \ell_n, \ell_{n+1} \in L_l \wedge 0 \leq n \leq 5\}$ as introduced in [KP10a, KP10b]. The lattice $(L_h, \leq_h)$ is a lattice with dominance being wide as shown in Figure 4.1. The last two lattices are built taking the structure of lattice $(L_h, \leq_h)$. They have ten elements instead of six elements. lattice $(L_g, \leq_g)$ and lattice $(L_t, \leq_t)$ are shown in Figure 4.2, respectively.

Using the algorithm presented in [BKP09], all boundaries for each consequence of the ontologies with respect to the lattices are computed and saved as input needed for the evaluation. For O^{SNOMED}, we use at most 100 consequences per label. Precisely, 100 consequences each with label ℓ_0, ℓ_1, ℓ_2, ℓ_3, ℓ_4, ℓ_5 which appear the same in every lattice, ℓ_8, ℓ_9 which appear in the lattice $(L_g, \leq_g)$ and

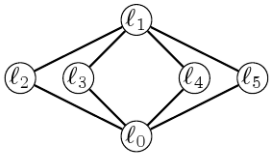


Figure 4.1: Lattice $(L_h, \leq_h)$.

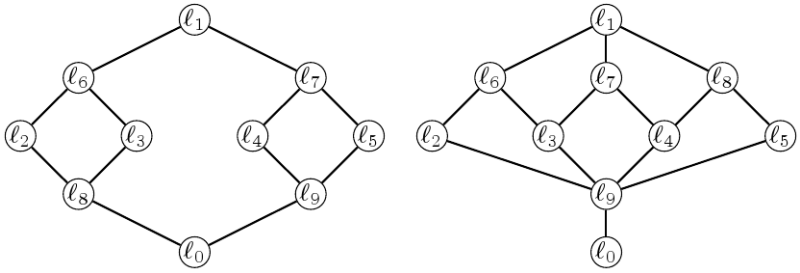


Figure 4.2: Lattice $(L_g, \leq_g)$ (left) and Lattice $(L_t, \leq_t)$ (right).

Condition	Goal label (ℓ_g)	Target label (ℓ_t)
$\leq$	$\forall \ell, \ell \not\geq \nu$	$= \ell_g$
$\geq$	$\forall \ell, \ell \not\leq \nu$	$= \ell_g$
$\not\leq$	$\forall \ell, \ell \geq \nu$	$\forall \ell, \ell \not\leq \ell_g$, limited to 3 labels
$\not\geq$	$\forall \ell, \ell \leq \nu$	$\forall \ell, \ell \not\geq \ell_g$, limited to 3 labels

Table 4.1: Label selected for each condition.

with label ℓ_7 only seven consequences which appear in the lattices $(L_g, \leq_g)$ and $(L_t, \leq_t)$. For O^{FUNCT}, we use at most 5 consequences per label. The test includes five consequences with boundary equal to ℓ_0, ℓ_1, ℓ_2, ℓ_3, ℓ_4, two consequences ℓ_5 which computed in all the lattices and five consequences ℓ_6, ℓ_7, ℓ_8, and ℓ_9 which computed using lattice $(L_g, \leq_g)$.

Using an ontology, a labeling lattice, a list of computed boundaries w.r.t. each consequence, a set of selected consequences, one test runs to compute the change sets for each consequence with each of conditions showing on the Table 4.1. For example, we have a consequence with boundary ℓ_3 for lattice $(L_d, \leq_d)$. For condition not greater than or equal to, the tests are changing the boundary to a value $\not\geq \ell_3$ and $\not\geq \ell_0$. In addition to each test, we combine them with at most 3 labels which are not greater than or equal to ℓ_g. For $\ell_g = \ell_3$, the target labels are labels ℓ_5, ℓ_0. In the case of changing the boundary to a value $\not\geq \ell_g$, if the ℓ_g is equal to the bottom element of the lattice, this computation is

eliminated, because there is no value which is smaller than the bottom element. The analogous way applies to the case of changing the boundary to a value $\not\leq \ell_g$, where ℓ_g is the top element of the lattice.

4.3 Evaluation Results

Our evaluation shows that our algorithm behaves well to find a minimal change set of a given consequence and a goal value with respect to the condition. That means for every consequence we obtain the correct boundary w.r.t the change sets computed. We show the results in comparison tables and graphs. In the figures, Ld stands for lattice $(L_d, \leq_d)$, Ll for lattice $(L_l, \leq_l)$, Lh for lattice $(L_h, \leq_h)$, Lg for lattice $(L_g, \leq_g)$, and Lt for lattice $(L_t, \leq_t)$. The scale of Figures 4.3 and 4.5 is the logarithmic scale.

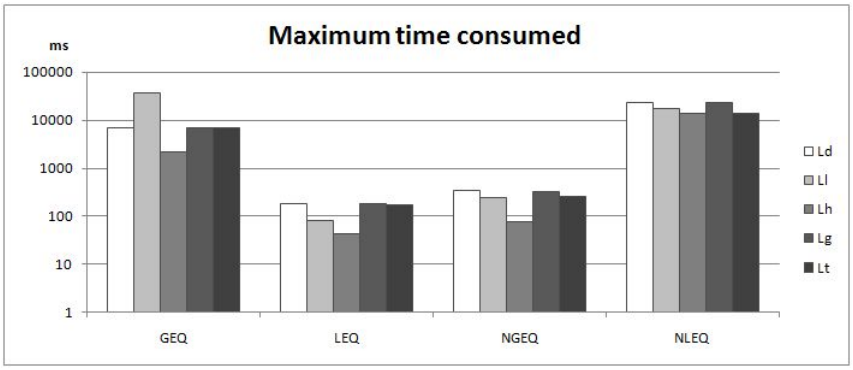
From all the consequences and all the changes we have in our test plan for O^{SNOMED}, the comparison of maximum time consumed to compute the minimal change set is shown in Figure 4.3 (a). For case of greater than or equal to, lattice $(L_l, \leq_l)$ has a consequence where it took 37,092 ms to change the consequence's label from ℓ_0 to ℓ_4. It was interrupted after it computed 10 minimal change sets. The cardinality of its minimal change set is found 9 axioms. What interesting is using not less than or equal to condition for this consequence w.r.t. the dual condition, we obtain 9 minimal change sets. Each of them has the smallest cardinality, i.e. 9 axioms. The computation took 10,310 ms. The results for case less-than-or-equal and case not greater than or equal to are obtained very fast i.e. from 42 ms to 337 ms. In all cases, the minimal time consumed is 0 ms.

Figure 4.3 (c) shows the average time consumed for all the consequences and conditions. The results for conditions less than or equal to and not greater than or equal to are extremely low, ranging from 3,02 ms to 7,58 ms with standard deviation 3,73 ms to 11,34 ms. On the other hand, the result for condition not less than or equal to is quite high, ranging from 39,36 ms to 78,19 ms with the highest standard deviation 919,74 ms for lattice $(L_l, \leq_l)$. This shows that the time consumed for repairing the boundary using lattice $(L_L, \leq_l)$ is spread in very big range. The average results for the condition greater than or equal to are very high and with high number of standard deviation.

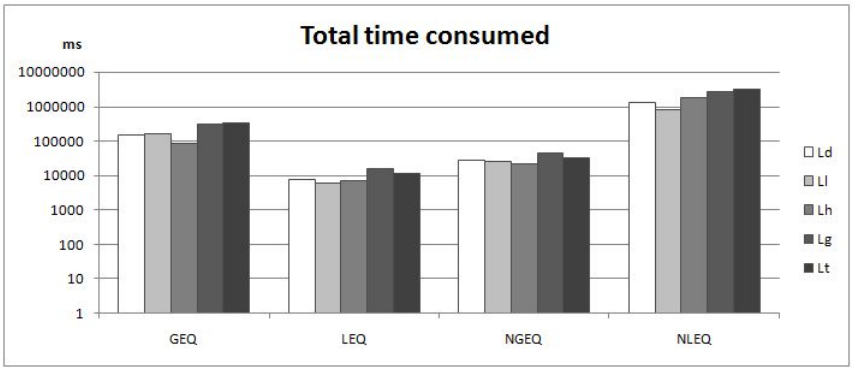
The total amount of time used in repairing the consequences varies in different cases. In case of changing the consequences to less than or equal to and not greater than or equal to, the computation took less than a minute for every lattice. The largest amount of time used to repair the boundary is on lattice $(L_t, \leq_t)$ with condition not greater than or equal to which is shown in Figure 4.3.

In Figure 4.4 (a), it is interesting that the largest cardinality of the minimal change set for case less than or equal to and not greater than or equal to is always one axiom. Figure 4.4 (b) shows that the average of cardinality of minimal change

(a)



(b)



(c)

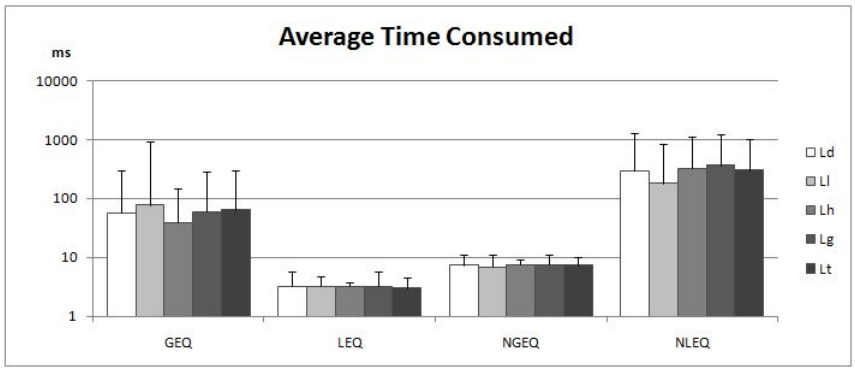
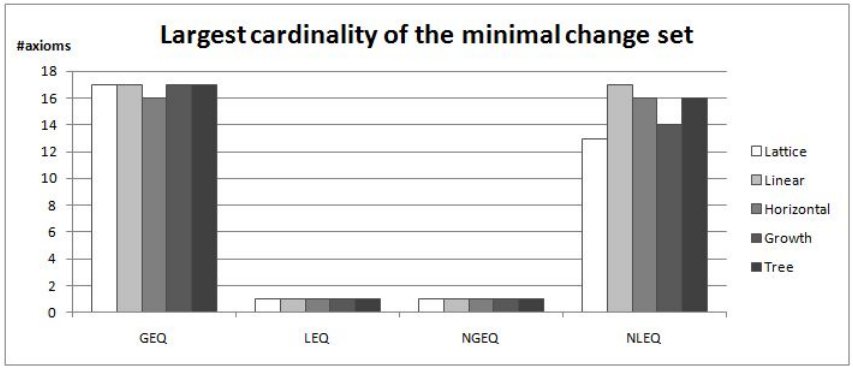


Figure 4.3: Maximum, total, average time consumed for boundary repairing for O^{Snomed}.

set for these cases is one. The standard deviation is zero in this case, since for all consequences, the cardinality of all minimal change set is one. For the rest of the cases, we obtained 13 axioms, 14 axioms, 16 axioms, and 17 axioms as the largest cardinalities of the minimal change set.

Figure 4.5 shows the comparison of maximum time, minimum time and average time consumed to compute the smallest minimal change set for each consequence in the test case in ontology O^{Funct}. Both less than or equal to and greater than or equal to cases are showing similar performance. They are in average 1 second per consequence's change. On the other hand, not less than or equal to

(a)



(b)

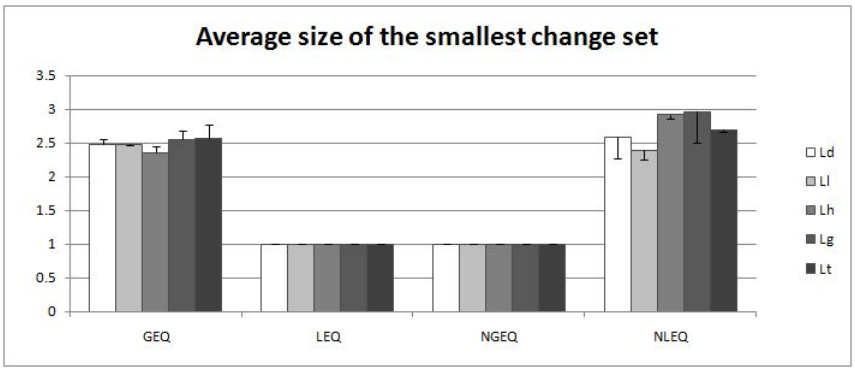


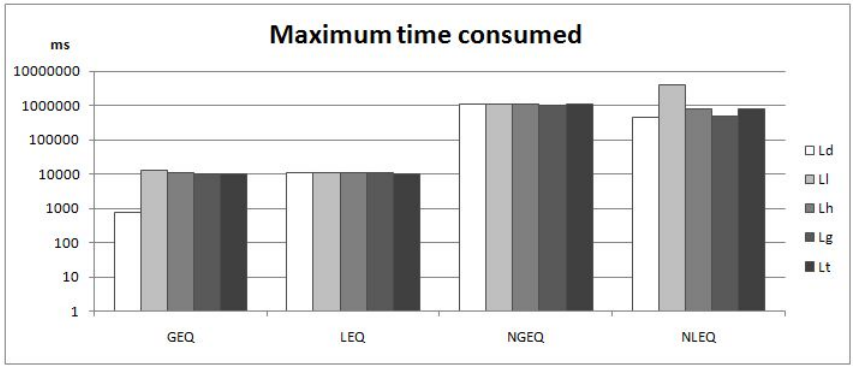
Figure 4.4: The largest cardinality and average size of the smallest change set of O^{SNOMED}.

and not greater than or equal to cases are taking more time which in average it computed the minimal change set in 100 seconds. Contrary from the change set computation of O^{SNOMED}, the minimal time is around 300 ms.

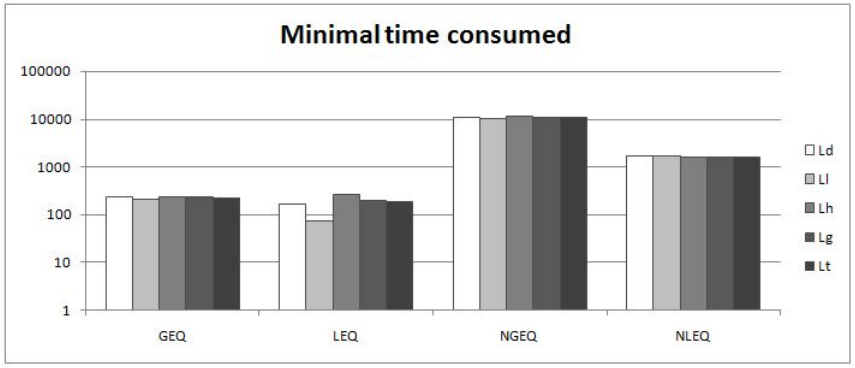
The number of the largest cardinality of the smallest minimal change set which is shown in Figure 4.6 (a) is 5 axioms in all cases, except for lattice $(L_g, \leq_g)$ in NGEQ and NLEQ conditions and lattice $(L_d, \leq_d)$ in NLEQ condition are 4 axioms. This number is relatively smaller than the number that we obtained for ontology O^{SNOMED}.

As we mentioned before, there are two conditions that will trigger the interruption. The first condition is when we computed a change set with cardinality one. The computation is interrupted since we have found the smallest and minimal change set. Any further computation will obtain only at most a change set with same size. The second condition is when we have computed 10 minimal change sets and still the hitting set tree algorithms has more branches to be extended. We say the interruption of cardinality a possitive interruption. In the sense, the more interruptions of cardinality one is better because the algorithm is terminated once the minimal change set computed. On the other hand the interruption of 10 minimal change sets is a negative interruption, where the more

(a)



(b)



(c)

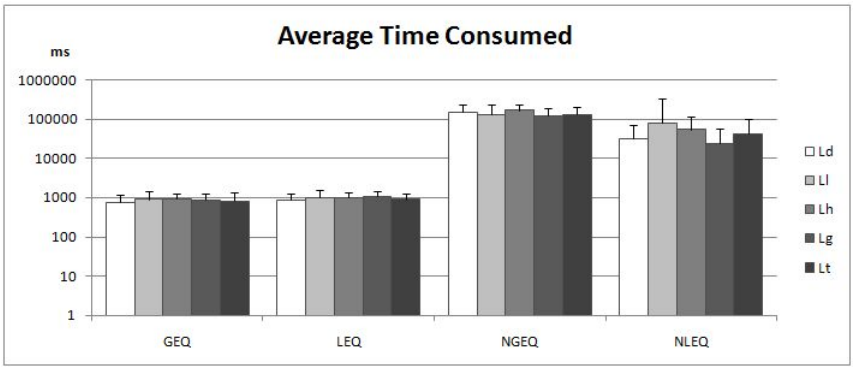


Figure 4.5: Maximum, minimal, average time consumed for boundary repairing for O^{FUNCT}.

computed is not because the result of the smallest minimal change set might not be the smallest minimal change set.

For the possitive conditions, we record only the interruption of more than 10 change sets computed, because the interruption of cardinality one CS was not implemented as the interruption of the part of the test, instead it is an optimization which is part of the HST algorithm in [KP10b]. The HST algorithm keeps updating the variable n which is an optimization to compute an IAS or RAS which is up to size n. So, when it has obtained a change set of size one, the

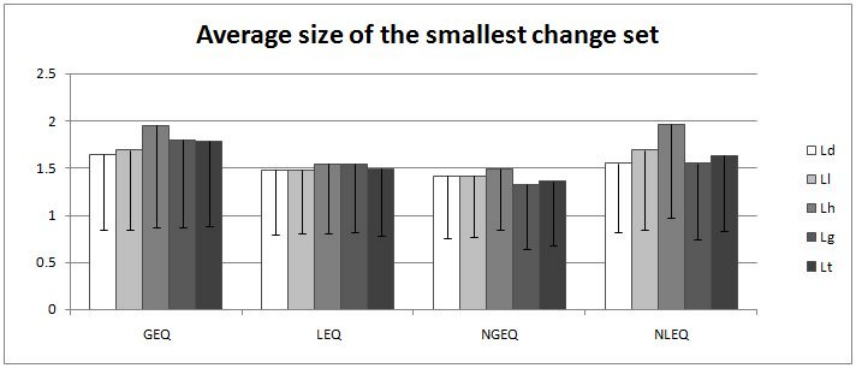


Figure 4.6: A average size of the smallest change set in repairing boundary of O^{Funct}.

Lattice	Case	#Consequence	%Interrupted Consequence	#Interruption >=10 CS	#Interruption |CS| = 1
$(L_d, \leq_d)$	GEQ	2614	1.38%	36	-
	LEQ	2328	0.00%	0	-
	NGEQ	3756	100.00%	0	3,756
	NLEQ	4514	48.25%	138	2,040
$(L_h, \leq_h)$	GEQ	3,100	1.48%	46	-
	LEQ	2,135	0.00%	0	-
	NGEQ	2,805	100.00%	0	2,805
	NLEQ	5,700	51.35%	437	2,490
$(L_l, \leq_l)$	GEQ	2,133	1.50%	32	-
	LEQ	1,902	0.00%	0	-
	NGEQ	3,798	100.00%	0	3,798
	NLEQ	4,426	52.94%	68	2,231
$(L_g, \leq_g)$	GEQ	5,320	1.65%	88	-
	LEQ	4,759	0.00%	0	-
	NGEQ	5,829	100.00%	0	5,829
	NLEQ	7,512	43.56%	407	2,865
$(L_t, \leq_t)$	GEQ	5,335	1,87%	100	-
	LEQ	3,756	0.00%	0	-
	NGEQ	4,370	100.00%	0	4,370
	NLEQ	10,521	54.98%	755	5,029

Table 4.2: O^{Snomed} interrupted comparison.

computation is stopped because it computed nothing with size smaller than one.

Table 4.2 shows interesting comparison of the numbers how many computations in O^{Snomed} were interrupted, i.e. the computations were terminated before they finished searching all the change sets. Condition not greater than or equal to in all the lattices is consistently showing 100% interruption and condition not less than or equal to is more or less 50% interruption appeared in the computation of change set. For ontology O^{Funct}, Table 4.3 shows less interruption in greater than or equal to and less than or equal to conditions than in the rest

Lattice	Case	#Consequence	%Interrupted Consequence	#Interruption >=10 CS	#Interruption |CS| = 1
$(L_d, \leq_d)$	GEQ	131	0.76%	1	-
	LEQ	154	1.30%	2	-
	NGEQ	273	90.11%	49	197
	NLEQ	212	63.21%	2	132
$(L_h, \leq_h)$	GEQ	118	0.85%	1	-
	LEQ	168	1.19%	2	-
	NGEQ	351	87.18%	72	234
	NLEQ	201	40.80%	3	79
$(L_l, \leq_l)$	GEQ	130	1.54%	2	-
	LEQ	105	0.95%	1	-
	NGEQ	201	83.58%	22	146
	NLEQ	263	52.85%	9	130
$(L_g, \leq_g)$	GEQ	256	1.17%	3	-
	LEQ	304	1.31%	4	-
	NGEQ	501	90.22%	71	381
	NLEQ	357	57.42%	0	205
$(L_t, \leq_t)$	GEQ	279	1.43%	4	-
	LEQ	279	1.08%	3	-
	NGEQ	348	91.67%	62	257
	NLEQ	432	56.71%	6	239

Table 4.3: O^{FUNCT} interrupted comparison.

		Time (ms)				
Ontology	Case	Max	Min	Avg	Stdev	Total
O^{SNOMED}	GEQ	1,761	0	50.28	181.53	20,111
	NLEQ	23,194	0	292.35	1,504.04	116,949
O^{FUNCT}	GEQ	6,190	373	964.61	1237.38	22,186
	NLEQ	437,487	1,741	38,105.78	98,304.39	876,433

Table 4.4: Time results comparison between $\geq$ and $\not\leq$ in O^{SNOMED} and O^{FUNCT} in Lattice $(L_d, \leq_d)$.

conditions. The percentage of interruption for not greater than or equal to is very high around 83% to 92% and for not less than or equal to is around 40% to 63%.

Based on the behaviour on the not greater than or equal to condition of having high percentage of interruption, the fact that most of the interruptions were triggered by obtaining a cardinality one change set, and the average size of the smallest minimal change set, namely one, we can derive that ontology O^{SNOMED} has for each consequences it has at least one diagnosis with cardinality one axiom and ontology O^{FUNCT} has for most consequences it has at least one diagnosis with cardinality one axiom. Therefore, when we have found a diagnosis with cardinality one axiom in a computation, it is for sure that the computation interrupted by condition a change set with cardinality one is found.

As the performances of all the algorithms are quite different, it is interesting to compare between two algorithms that can fulfill a certain request. Tables 4.4

Ontology	Case	Size (axiom) Max	Min	Avg	Stdev	#Diff ν'
O^{SNOMED}	GEQ	12	1	2.5	2.15	0
	NLEQ	12	1	2,51	2,16	
O^{FUNCT}	GEQ	4	1	1.70	0.88	2
	NLEQ	4	1	1.70	0.88	

Table 4.5: Size results comparison between $\geq$ and $\not\leq$ in O^{SNOMED} and O^{FUNCT} in Lattice $(L_d, \leq_d)$.

and 4.5 compare the performance of the greater than or equal to condition with the not less than or equal to condition for O^{SNOMED} and O^{FUNCT}. We collect the data of changing the boundary of consequences with goal label which is greater than or equal to ℓ_5 for lattice $(L_d, \leq_d)$. Either using not less than or equal to ℓ_2 or greater than or equal to ℓ_5, the range of boundary is the same, namely ℓ_1, ℓ_4, ℓ_5. There are 400 consequences in O^{SNOMED} and 23 consequences in O^{FUNCT}. In the computation using condition $\geq$, the target label has a fixed label, i.e. equal to the goal label, but for condition $\not\leq$, it does not. Thus, we collected only ones with the precomputed boundary ℓ_0, ℓ_2, and ℓ_3 and the target label ℓ_5.

Table 4.4 shows the amount of time used in the computation of boundary repairing. Using greater than or equal to condition is much faster than using not less than or equal to condition. For O^{SNOMED}, the positive condition (greater than or equal to) is six times faster than the negative condition (not less than or equal to) for the total time used. For O^{FUNCT}, the positive condition is 39 times faster than the negative condition. Table 4.5 shows that the results are similar in both cases w.r.t. the ontology. From the computed label of the consequences w.r.t. the relabeling of the change set, there are only two consequences with different ν' in ontology O^{FUNCT} and zero in ontology O^{SNOMED}. The two consequences with different ν' both obtained ℓ_4 in the not less than or equal to case and ℓ_1 in the greater than or equal to case. This could happen when we have different smallest change sets. We believe the value of boundary which is closer to the goal label is better than the one which is far from it, because that means the change on the labels of the axioms is not so much far from the original labels of the axioms. Although we get 2 consequences with better new boundary using the change set from $\not\leq \ell_5$, the overall computation using condition $\geq \ell_2$ is better than the one using condition $\not\leq \ell_5$.

Chapter 5

Conclusions

In this thesis, we have shown that it is possible to give general conditions for the repairing of the label of consequence or boundary. Possible conditions that have been introduced here are greater than or equal to, less than or equal to, not greater than or equal to, and not less than or equal to. The possibility of changing the boundary with a given condition depends on the given goal label. We can not repair the value of a boundary to be greater than the top element of the lattice, or not to be less than the bottom element of the lattice. The first two cases can be addressed using the methods introduced in [KP10b, KP10a]. Solutions for the last two cases are introduced in Chapter 3. The solutions are mainly finding which axioms that need to be relabeled in order to change the label of the consequence.

The minimal change set w.r.t. the possitive conditions is an IAS or an RAS. The basic ideas are taken from a minimal axiom set and a diagnosis of the ontology and the given consequence. From these basic ideas, a minimal change set is a sub-ontology $\mathcal{O}_{\not\geq \ell_g}$ for IAS or $\mathcal{O}_{\not\leq \ell_g}$ for RAS which contains axioms of a MinA or a diagnosis, respectively.

The same idea was taken in order to solve the negative conditions. A MinA is a change set for changing a boundary to a value $\not\leq \ell_g$ and a diagnosis is a change set for changing a boundary to a value $\not\geq \ell_g$. Then, from these basic ideas, we tried to apply the same like in the positive conditions. In those cases, the minimal change set is a subset of a MinA (or a diagnosis) whose axioms is labeled not greater than or equal to (or not less than or equal to) the given goal label. Unfortunately, for the negative ones, we cannot just select all axioms which have labels greater than or equal to (or less than or equal to) the given goal label, because any subset of a diagnosis or a MinA that has the supremum or infimum greater than or equal to or less than or equal to the goal label is needed to be split until the supremum or infimum not greater than or equal to or not less than

or equal to the goal label. All the labels of part of the subset is contained in a solution label set. From this set we can extract the axioms that needs to be relabeled, i.e. the axioms which are.

The result of the empirical evaluation shows that the implementations of case the possitive conditions perform well in the computation of the smallest minimal change set which is obtained from computing the partial IAS and RAS and the implementation of negative conditions take more time in the computation of the smallest minimal change set which is obtained from computing the solution label set and the solution set.

With the comparison of the case of not less than or equal to ℓ_2 and greater-than-or-equal to ℓ_5 using lattice $(L_d, \leq_d)$, we see big difference between two different types of algorithms. The algorithms for not less than or equal to and not greater than or equal to cases underperform. They need more time in obtaining the smallest change sets, but occasionaly they could give some different new boundaries which have better values because it means the axioms of the change sets are labeled to some values closer to the original labels.

For future work, one could consider how to optimize the algorithms for negative conditions such that their performances can compete with the performance of algorithms which are used to compute change set for less than or equal to and greater than or equal to cases. One could analyze if the optimization $n - 1$ branching size for n is the size of node. Another idea is how to use the algorithms of the possitive conditions to give a solution for a negative condition. It could be done by analyzing the range that negative conditions apply. By seeing what is the range given by the negative conditions, we can build an approach using the positive conditions to cover the same range. Similar idea was used in the comparison between boundary changes to a value $\geq \ell_5$ and to a value $\not\leq \ell_2$. It was shown that we can use the algorithm to compute the smallest IAS to obtain a solution for changing the boundary to a value $\not\leq \ell_2$.

Bibliography

[BCM+03] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003. [cited at p. 5]

[BKP09] 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 *ISWC '09: Proceedings of the 8th International Semantic Web Conference*, volume 5823 of *LNCS*, pages 49–64. Springer-Verlag, 2009. [cited at p. vii, 2, 5, 6, 7, 8, 9, 10, 11, 36, 38]

[BLM+05] F. Baader, C. Lutz, M. Milicic, U. Sattler, and F. Wolter. A description logic based approach to reasoning about web services. In *Proceedings of the WWW 2005 Workshop on Web Service Semantics (WSS2005)*, Chiba City, Japan, 2005. [cited at p. 1]

[BP10a] F. Baader and R. Peñaloza. Automata-based axiom pinpointing. *Journal of Automated Reasoning*, 45(2):91–129, August 2010. Special Issue: Selected Papers from IJCAR 2008. [cited at p. 9]

[BP10b] F. Baader and R. Peñaloza. Axiom pinpointing in general tableaux. *Journal of Logic and Computation*, 20(1):5–34, 2010. Special Issue: Tableaux and Analytic Proof Methods. [cited at p. 9]

[BPS07] F. Baader, R. Peñaloza, and B. Suntisrivaraporn. Pinpointing in the description logic $\mathcal{EL}$. In *Proceedings of the 30th German Conference on Artificial Intelligence (KI2007)*, volume 4667 of *Lecture Notes in Artificial Intelligence*, pages 52–67, Osnabrück, Germany, 2007. Springer-Verlag. [cited at p. 9]

[DP02] B. A. Davey and H. A. Priestley. *Introduction to Lattices and Order*. Cambridge University Press, second edition, 2002. [cited at p. 7]

[GKL09] A. Gaag, A. Kohn, and U. Lindemann. Function-based solution retrieval and semantic search in mechanical engineering. In *Proceedings of the International Conference on Engineering Design (IDEC2009)*, 2009. [cited at p. iii, 38]

[KP10a] M. Knechtel and R. Peñaloza. Correcting access restrictions to a consequence. In Volker Haarslev, David Toman, and Grant Weddell, editors, *Proceedings*

of the 2010 International Workshop on Description Logics (DL2010), volume 573 of *CEUR-WS*, 2010. [cited at p. 3, 5, 6, 8, 14, 15, 16, 19, 31, 38, 47]

[KP10b] M. Knechtel and R. Peñaloza. A generic approach for correcting access restrictions to a consequence. In Lora Aroyo, Grigoris Antoniou, Eero Hyvönen, Annette ten Teije, Heiner Stuckenschmidt, Liliana Cabral, and Tania Tudorache, editors, *Proceedings of the 7th Extended Semantic Web Conference (ESWC 2010)*, volume 6088 of *Lecture Notes in Computer Science*, pages 167–182, 2010. [cited at p. 3, 5, 6, 8, 14, 15, 16, 17, 18, 19, 24, 31, 38, 43, 47]

[KPHS07] A. Kalyanpur, B. Parsia, M. Horridge, and E. Sirin. Finding all justifications of OWL DL entailments. In *ISWC/ASWC*, pages 267–280, 2007. [cited at p. 36]

[Rei87] R. Reiter. A theory of diagnosis from first principles. *Artif. Intell.*, 32(1):57–95, 1987. [cited at p. 6, 9, 10]

[Rym92] R. Rymon. Search through systematic set enumeration. In *KR*, pages 539–550, 1992. [cited at p. 6]

[SBF98] R. Studer, V. R. Benjamins, and D. Fensel. Knowledge engineering: Principles and methods, 1998. [cited at p. 5]

[SC03] S. Schlobach and R. Cornet. Non-standard reasoning services for the debugging of description logic terminologies. In *Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI)*, pages 355–362, 2003. [cited at p. 1, 9]

[sno] SNOMED CT ontology. http://www.ihtsdo.org/snomed-ct/. [cited at p. iii, 38]