Master's Thesis

Iterative Ontology Update with Minimum Change

by

Aparna Saisree Thuluva

in partial fulfilment of the requirements

for the degree of Master of Science

in

International Center for Computational Logic

Faculty of Computer Science

Technische Universität Dresden

Supervisor:
Prof. Dr.-Ing. Franz Baader

Advisor:
Dr. rer. nat. Rafael Peñaloza

Dresden, April 2015

Declaration of Authorship

Hereby, I certify that this dissertation titled "*Iterative Ontology Update with Minimum Change*" is the result of my own work and is written by me. Any help that I have received in my research work has been acknowledged. I certify that I have not used any auxiliary sources and literature except those cited in the thesis.

Author	:	Aparna Saisree Thuluva
Matriculation number	:	3933381
Date of submission	:	03.03.2015
Signature	:	

"We can't solve the problems by using the same kind of thinking we used when we created them"

Albert Einstein

Abstract

Ontologies, which form the core of Semantic Web systems, need to evolve to meet the changing needs of the system and its users. The dynamic nature of ontology development has motivated the formal study of ontology evolution problems, which is one of the important problems in the current Semantic Web research. Ontology evolution approaches suffer from intrinsic information loss. The current study deals with the problem of minimizing information loss during iterative ontology update. It provides a framework combining the ontology evolution tasks with context-based reasoning method. Using this framework, all the solutions obtained in an ontology evolution task, which are partly redundant, can be described as contexts and compactly represented in a single labelled ontology. Further updates and reasoning can be done on this ontology efficiently.

We propose new approaches to do ontology contraction, ontology expansion and ontology revision using context-based reasoning method. These approaches show how ontology evolution can be done with minimum information loss by using all the solutions obtained at every stage of the evolution task, efficiently using our framework. We also propose theoretical methods to extract the optimal solutions from the ontology obtained as the result of iterative ontology update. We show that, optimal solutions in the intermediate stages of iterative ontology update may not be the optimal solutions in the result obtained at the end of all the stages. We handle various notions of an optimal solution: the solution which changes the semantics of the ontology as minimum as possible, the solution which contains some of the intended consequences, the solution which has the most original axioms of the ontology. We also propose theoretical methods to do context-based reasoning over the optimal solutions extracted.

We present the first prototypical implementation of the theoretical methods developed in this thesis and show the preliminary results of our implementation on the real-world ontologies.

Acknowledgements

I would like to express my heartfelt thanks to Dr. Rafael Peñaloza, my thesis advisor, for his ceaseless support and encouragement during my thesis and project. He is the one who inspired me towards research. He is the true personification of an ideal supervisor. His passion and focus towards research always motivated me and gave me a new energy to do my thesis. I would like to thank Prof. Franz Baader for giving me an opportunity to do my thesis in his research group.

I would like to thank Prof. Steffen Hölldobler, the director of International Master program in Computational Logic for encouraging me and supporting me by granting tuition fee waiver for four semesters. I would like to express my gratitude to Dr. Anni Yasmin Turhan for her support and encouragement. I would like to thank all my professors and secretaries Sylvia, Julia and Kerstin for continuously helping me.

My special thanks to all my friends in Dresden, who not only made my stay in Dresden very memorable, but also helped me a lot in studies during the initial days of my masters. Without their support, I would not have be so successful in my masters.

I finally want to thank my family and friends, who are the most important people in my life; my husband Sainitin, my parents Gopal and Vijayalakshmi and my brother Aditya. I deeply thank them all for making me what I am today and for always being my source of energy. My special thanks to my husband for his unconditional love and support. He is the one who encouraged me to pursue masters and he is the man behind my success. My deepest gratitude to my parents, they always believed in me and supported me during my hard times. It is only because of their support, I am in Dresden today and finished my masters successfully.

Contents

Dedicated to my family and my teachers...

Chapter 1

Introduction

Description Logics (DL) [1] are a family of knowledge representation formalisms with well-understood computational properties. Description Logics can be used to represent the conceptual knowledge of an application domain in a structured and formally well-understood way. From the DL point of view, an ontology is a finite set of axioms, which formalize our knowledge about the concepts of an application domain. The Web Ontology Language (OWL) [2] which is the standard ontology language for the semantic web is a DL-based language. Ontologies are the main driving force behind the semantic web vision. They are complex in nature, often large structured and are not static entities. They are frequently modified when new information needs to be incorporated, or existing information is no longer considered to be valid. This dynamic nature of ontologies motivates the study of *ontology evolution* problems [3, 4].

1.1 Motivation

Ontology Evolution deals with the growth of an ontology. More specifically, ontology evolution means modifying or upgrading an ontology when there is certain need for change or there comes a change in the domain knowledge. The main tasks in ontology evolution are: ontology contraction, ontology expansion and ontology revision. Ontology contraction is the process of "retracting" a consequence from an ontology that is no longer allowed to hold. The process of "adding" a new axiom to an ontology is called ontology expansion and the process of "adding" a new axiom to an ontology while ensuring the consistency of the ontology is called ontology revision. Ontology expansion does not ensure the consistency of the ontology after adding the axiom.

The evolution of an ontology O results in another ontology O' in which the required information (consequence) is incorporated, retracted or updated. All the ontology evolution tasks should satisfy the *principle of minimal change* [5], according to which the semantics of the ontology should change "as little as possible", thus ensuring that evolution has the least possible impact on the ontology. An ontology evolution task should satisfy all the AGM [6]

postulates specified for that evolution task. Consider the task of ontology contraction, in practice the approaches adopted to ontology contraction are *syntactic approaches* [7], which are based on the notion of a *justification:* a minimal subset of the ontology that entails a given consequence [8, 9]. To retract a consequence α from an ontology O, it suffices to compute all the justifications for α w.r.t the ontology O and then compute a maximal subset R of O (a repair) with at least one axiom from each justification and the ontology $O' = O \setminus R$ is the result of the contraction task. This solution complies with a syntactical notion of minimal change as retracting α results in the deletion of a minimal set of axioms from O and hence the structure of O is maximally preserved [7]. There exists many solutions for ontology contraction task. But, usually the solution which satisfies the principle of minimal change is chosen as the optimal solution. However, these practical approaches suffer from inherent information loss. More precisely, by removing R from O, we may retract axioms of O other than α, which are "intended". Identifying and recovering such intended consequences is an important issue [7]. The current study addresses this issue of minimising the information loss during iterative ontology update. Consider the following example.

Example 1.1. *Consider an ontology with the following axioms.*
O=$\{A \sqsubseteq B,\ B \sqsubseteq C,\ B \sqsubseteq E,\ C \sqsubseteq E,\ E \sqsubseteq F\ \}$.
The task is to do iterative ontology contraction on O with respect to the consequences $A \sqsubseteq F$ *and* $B \sqsubseteq E$.
Step 1: *To retract the consequence* $A \sqsubseteq F$ *from O, it suffices to compute all the justifications for* $O \models A \sqsubseteq F$. *The* ***justifications*** *for this entailment are:*
$J_1 = \{A \sqsubseteq B,\ B \sqsubseteq E,\ E \sqsubseteq F\}$,
$J_2 = \{A \sqsubseteq B,\ B \sqsubseteq C,\ C \sqsubseteq E,\ E \sqsubseteq F\}$.
By removing atleast one axiom from each justification we get many solutions (repairs) for the contraction, but the following three solutions are interesting for us.

The possible solutions for this contraction are:
$S_1 = \{B \sqsubseteq C,\ B \sqsubseteq E,\ C \sqsubseteq E,\ E \sqsubseteq F\}$ *(by removing the axiom* $A \sqsubseteq B$ *from* J_1, J_2*),*
$S_2 = \{A \sqsubseteq B,\ B \sqsubseteq C,\ B \sqsubseteq E,\ C \sqsubseteq E\}$ *(by removing the axiom* $E \sqsubseteq F$ *from* J_1, J_2*),*
$S_3 = \{A \sqsubseteq B,\ B \sqsubseteq C,\ E \sqsubseteq F\}$ *(by removing the axioms* $B \sqsubseteq E,\ C \sqsubseteq E$ *from* J_1, J_2*).*

Among these solutions only two of them satisfy the principle of minimal change. They are: S_1 *and* S_2*, as they retract minimal set of axioms from O, they are the optimal solutions. As it requires more memory to store all the solutions and more time to do computations on all the optimal solutions, generally only one optimal solution is chosen as the result of ontology update. Let the optimal solution chosen from this contraction be* S_1.
Step 2: *To do iterative ontology update, the next consequence* $B \sqsubseteq E$ *should be retracted from the solution obtained from the first contraction, that is* $B \sqsubseteq E$ *should be retracted from* S_1. *There exists two* ***justifications*** *for* $S_1 \models B \sqsubseteq E$. *They are:*
$J_1 = \{B \sqsubseteq E\}$,

$J_2 = \{B \sqsubseteq C,\ C \sqsubseteq E\}$.
By removing atleast one axiom from every justification we obtain two ***possible solutions****.*

> *The two possible solutions are:*
> $S_4 = \{C \sqsubseteq E,\ E \sqsubseteq F\}$ *and*
> $S_5 = \{B \sqsubseteq C,\ E \sqsubseteq F\}$.

Both the solutions are optimal as they satisfy the principle of minimal change. Both the solutions, S_4 *and* S_5 *remove* ***three axioms*** *from* O*. Thus either* S_4 *or* S_5 *can be chosen as the optimal solution for this contraction.*

From the above example, consider the solution S_3 obtained in first step, $S_3 \not\models A \sqsubseteq F$ and $S_3 \not\models B \sqsubseteq E$, that is, S_3 retracts both the consequences $A \sqsubseteq F$ and $B \sqsubseteq E$ from O and it removes only **two axioms** from O where as, the solutions S_4 and S_5 remove **three axioms** from O. That is, S_3 is the optimal solution for the iterative ontology contraction process on O. But, as S_3 is not the optimal solution in the intermediate step of the ontology contraction, it is not chosen in further steps. By selecting the solution S_1 after the first step and doing further contraction on this solution, we are losing some valuable axioms which are intended. From this example we show that, **in iterative ontology update optimal solution in the intermediate steps of ontology evolution may not give optimal solution at the end of all steps of ontology evolution process**.

1.2 Aim

Our goal is to do iterative ontology update with minimum information loss. As shown in the previous example, optimal solution in intermediate steps of ontology evolution may not be the optimal solution after all the steps of ontology evolution. Thus, to achieve minimum information loss it is very important to store all the solutions obtained in every step and do further updates on all the solutions. To achieve this goal, we should be able to compactly store all the solutions (which are partly redundant sub-ontologies) obtained in the intermediate steps of ontology evolution and do further computations on them in reasonable space and time. To do this, we introduce a new framework by combining ontology evolution approaches with context-based reasoning [10, 11] methods.

Context-based Reasoning [10, 11] A Context is a set of axioms. In context-based reasoning, an ontology is regarded as a set of contexts. There are two steps in context-based reasoning. The first step is to create a labelled ontology by annotating axioms of the given ontology with labels which are assigned to the contexts of the ontology. Each sub-ontology of an ontology can be defined as a context and every context is assigned with a label. Every axiom of the ontology that belongs to a context is annotated with the label assigned to that context, thus all the axioms of the ontology are labelled according to the contexts the axioms belong to. Thus all the sub-ontologies of an ontology are compactly represented as

single ontology with labelled axioms, which has less memory usage than storing each partly redundant sub-ontology separately.

The second step is to do context-based reasoning over the labelled ontology, this is the main aim of context-based reasoning method. Given a labelled ontology and a consequence, context-based reasoning checks the entailment of the consequence with respect to every context of the ontology. The result of context-based reasoning is called *Boundary* which is the list of labels of the contexts which entail the consequence.

1.3 Scope

The scope of the present work is to develop theoretical methods to handle ontology evolution tasks: ontology contraction, ontology expansion and ontology revision using context-based reasoning approach, theoretical methods to extract the optimal solutions from the ontology obtained as the result of doing ontology evolution tasks using context-based reasoning approaches and methods to do context-based reasoning over the extracted optimal solutions. We develop the first prototypical implementation for all the theoretical methods described and present the results of experiments performed on the benchmark ontologies SNOMED CT and FULL-GALEN, to evaluate our hypothesis that, our framework achieves the goal of iterative ontology update with minimum change.

The remaining chapters in this thesis are organized as follows 1.1. Chapter 2 presents the background details of context-based reasoning method and ontology evolution operations. It introduces all the basic definitions and notions that will be used in the rest of the thesis. All the definitions of Context-based reasoning and the method to label the axioms of an ontology to do ontology evolution tasks are presented. We also describe in detail, how context-based reasoning can done on the labelled ontologies. We present the definitions of ontology evolution tasks and present the existing practical approaches to do ontology evolution tasks.

Chapter 3 describes the theoretical methods to handle ontology evolution operations using context-based reasoning and presents the algorithms to do context-based ontology contraction, context-based ontology expansion and context-based ontology revision operations. It presents the relationship between ontology evolution operations and context-based ontology evolution operations. It gives examples to show how to do context-based ontology evolution operations and presents the theorems which prove the correctness and complexity of the algorithms developed to do context-based ontology evolution operations.

Chapter 4 focuses on the tasks extraction of the optimal solutions from a labelled ontology and methods to do context-based reasoning over the extracted optimal solutions. It also discusses about the notion for the best. There are many notions for the best to extract a optimal solution, for example, a solution with highest number of axioms, or a solution which entails a certain consequence. It also describes theoretical methods to

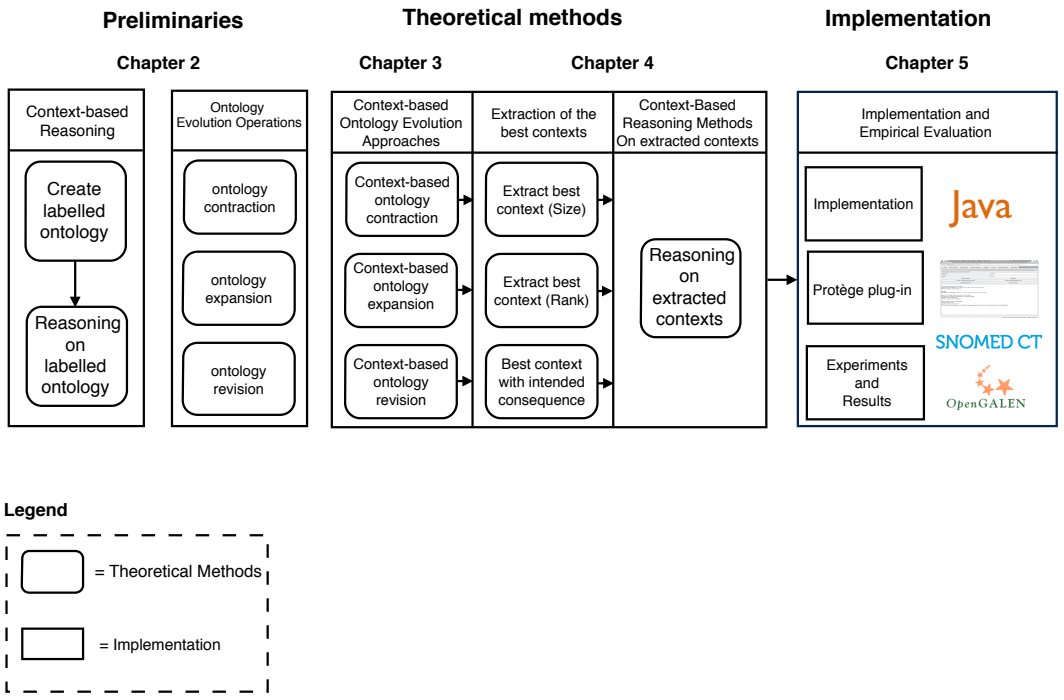

Figure 1.1 The flowchart represents the organization of theoretical and implementation work flow in this thesis.

extract optimal solutions and to do context-based reasoning over the optimal solutions extracted.

Chapter 5 presents the first prototypical implementation of the theoretical methods developed in Chapters 3 and 4. It presents the protége plug-in developed to do context-based ontology evolution operations. This chapter also presents the results of the experiments conducted and gives the interpretation of the results.

Chapter 6 discusses our contributions and limitations of our work, points to the direction for future work and summarizes this thesis.

Chapter 2

Preliminaries

This chapter introduces all the definitions and notions that are used in the rest of this thesis. Section 2.1 introduces all the definitions and basic details about context-based reasoning method. It describes the procedure to create a labelled ontology and the procedure to do context-based reasoning over a labelled ontology. Section 2.2 presents the definitions of ontology evolution operations: ontology contraction, ontology expansion and ontology revision.

2.1 Context-based Reasoning

Context-based reasoning [10] is a two step process. The first step is to label the axioms of the ontology with contexts, using a labelling function. The second step is to do reasoning over the labelled ontology. To do ontology evolution tasks using context-based reasoning, in this work we consider a special case of context-based reasoning where every repair for a consequence is described as a context in the ontology.

2.1.1 Label Ontology

In this thesis, we use a ontology model that regards an ontology as a finite set of general concept inclusion axioms (GCIs) of the form $L \sqsubseteq M$, where L, M are concept names. We further assume the existence of a binary relation $\models$ between an ontology and a consequence such that, for an arbitrary ontology K and an arbitrary consequence ψ, it holds that, if $K_1 \subseteq K$ and $K_1 \models \psi$ then $K \models \psi$.

The first step in context-based reasoning method is to describe contexts in the ontology. A context is a set of axioms or a sub-ontology of a given ontology. In general, the set of contexts in an ontology can describe the level of expertise of the user or the access restrictions for the user of the ontology. All the set of contexts are partly redundant sub-ontologies of an ontology. Storing all the contexts separately and doing reasoning over all the contexts is very expensive. This problem is handled in context-based reasoning by

assigning a label to every context of the ontology and creating a labelled ontology [10] by annotating every axiom of the ontology according to the context it belongs to. Thus, all the axioms of the ontology are labelled axioms where a label represents a context of the ontology.

Given an arbitrary ontology O and set of labels L, we assume that, every axiom $a \in O$ is assigned with a label $lab(a) \subseteq L$. Every label $l \in L$ is used to define a context in the ontology O. The sub-ontology accessible for the context with label l is defined to be

$$O_l = \{a \in O \mid l \in lab(a)\}.$$

Definition 2.1 (labelled ontology)**.** Given an ontology O and set of labels L, every axiom $a \in O$ is assigned a label $lab(a) \subseteq L$, which expresses the contexts from which the axiom a can be accessed. An ontology extended with such a labelling function lab is called a *labelled ontology*.

In this thesis, we consider a special case of labelled ontologies where every context is a repair for a given consequence w.r.t the given ontology. Thus all the repairs for a consequence will be described as contexts in the ontology. Given an arbitrary ontology O and a consequence α, a *repair* is defined as follows.

Definition 2.2 (repair)**.** A sub-ontology $R \subseteq O$ is a *repair* for $O \models \alpha$, if $R \not\models \alpha$ and for every $R' \subseteq O$ such that $R \subset R'$, it holds that $R' \models \alpha$.

Thus, a repair has two properties: it is maximal (w.r.t set inclusion) and it does not entail the given consequence α. The following gives an example of a repair.

Example 2.3. *Consider the ontology O shown in Figure 2.1. The ontology contains the following axioms:*
$a_1 : A \sqsubseteq B$
$a_2 : B \sqsubseteq C$
$a_3 : C \sqsubseteq D$
$a_4 : D \sqsubseteq E$
$a_5 : B \sqsubseteq E$
$a_6 : E \sqsubseteq F$
Let $\alpha = A \sqsubseteq F$ be the consequence for which the repairs should be computed w.r.t to the ontology O. A repair for $A \sqsubseteq F$ w.r.t the ontology O is the maximal sub-ontology of O which do not entail the consequence $A \sqsubseteq F$.

There are five repairs for $A \sqsubseteq F$ w.r.t O.
$R_1 = \{a_2, a_3, a_4, a_5, a_6\}$,
$R_2 = \{a_1, a_3, a_4, a_6\}$,
$R_3 = \{a_1, a_2, a_4, a_6\}$,
$R_4 = \{a_1, a_2, a_3, a_6\}$,
$R_5 = \{a_1, a_2, a_3, a_4, a_5\}$.
All the repairs are represented in the Figure 2.2.

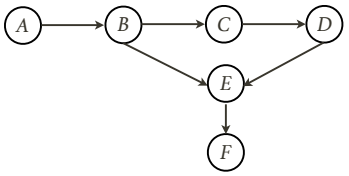

Figure 2.1: An unlabelled ontology

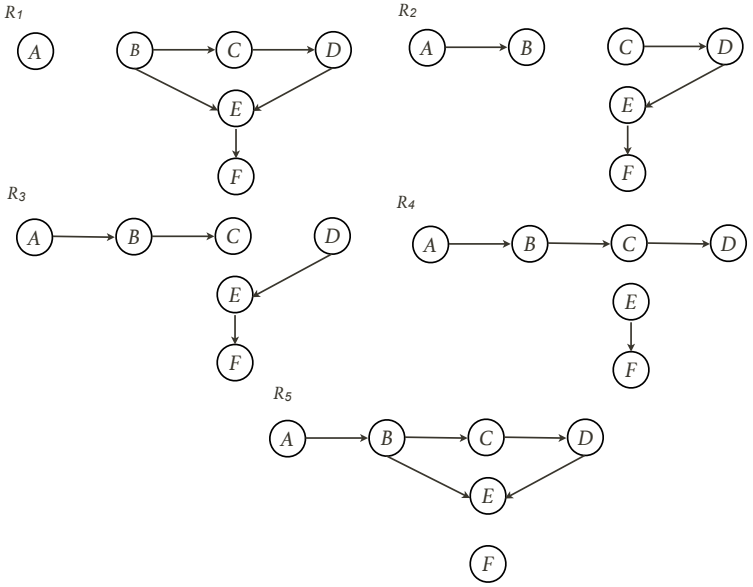

Figure 2.2: All repairs for the consequence w.r.t the ontology

R_1, R_2, R_3, R_4 and R_5 are repairs as they satisfy the following properties. Consider the repair R_1, $R_1 \not\models A \sqsubseteq F$ and R_1 is maximal w.r.t the ontology O. That is, for any axiom $a \in \{O \setminus R_1\}$, $R_1 \cup \{a\} \models A \sqsubseteq F$. Similarly the repairs R_2, R_3, R_4, R_5 also satisfy the above two properties and there exists no other repairs for $A \sqsubseteq F$ w.r.t O.

To compute repairs for a consequence, it is important to find the axioms responsible for the entailment. To find this, the technique of *axiom-pinpointing* [12, 13] is used, which is the task of finding the minimal sub-ontologies that entail a given consequence [10] (MinAs), or dually, the minimal sets of axioms that need to be removed or repaired to avoid deriving the consequence (diagnoses).

Definition 2.4 (MinA, diagnosis)**.** The set of axioms S is a *MinA* for $O \models \alpha$, if $S \subseteq O$ and $S \models \alpha$, for every $S' \subset S$, it holds that $S' \not\models \alpha$. A *diagnosis* for $O \models \alpha$ is a sub-ontology $S \subseteq O$ such that $O \backslash S \not\models \alpha$ and $O \backslash S' \models \alpha$ for every $S' \subset S$.

The sets of MinAs and diagnoses are dual in the sense that, from the set of all the MinAs, it is possible to compute the set of all diagnoses and vice versa through a hitting

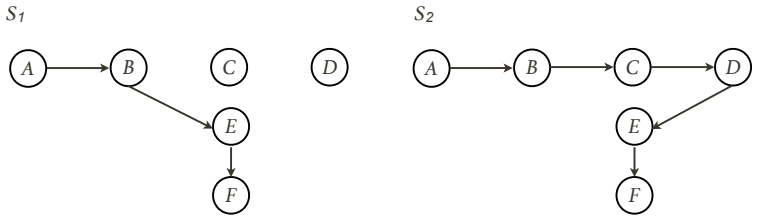

Figure 2.3: MinAs for the consequence w.r.t the ontology

set tree computation [14]. The following example shows the MinAs for a consequence w.r.t an arbitrary ontology.

Example 2.5. *Consider the ontology O shown in Figure 2.1 and the consequence* $\alpha = A \sqsubseteq F$.

There exists two MinAs for $O \models A \sqsubseteq F$. *They are,*
$S_0 = \{a_1, a_5, a_6\}$,
$S_1 = \{a_1, a_2, a_3, a_4, a_6\}$.
The MinAs S_0, S_1 *are represented in the Figure 2.3.*

S_0 *and* S_1 *are the MinAs for* $O \models A \sqsubseteq F$, *as they satisfy the following properties,* S_0 *and* S_1 *entail the consequence* $A \sqsubseteq F$, S_0 *and* S_1 *are minimal w.r.t O. That is, for any arbitrary set of axioms,* $S'_0 \subset S_0$, $S'_0 \not\models A \sqsubseteq F$.

In the above example, note that minimality is considered w.r.t set inclusion and not w.r.t the cardinality of a MinA. Diagnoses can be computed from the MinAs of a consequence. Given a diagnosis D for $O \models \alpha$, a repair R for $O \models \alpha$ is computed as complement of D i.e., $R = O \setminus D$. The following section describes the procedure to compute diagnoses from the MinAs and procedure to compute the repairs for a consequence w.r.t to an ontology using hitting set tree based approach for axiom-pinpointing.

Computing Repairs Based On Reiter's Hitting Set Tree Algorithm

Here we present a black-box algorithm to compute repairs for a consequence w.r.t to a given ontology, which is a variant of the Hitting-Set-Tree-based method [14] (HST-approach) for axiom pinpointing [12, 13]. We use black-box approach as it behaves well in practice with modern reasoners. It uses an un-modified reasoner and is independent of the specific logic or consequence as long as the reasoner exists. This HST-based method gets one MinA at a time from the black-box reasoner for $O \models \alpha$, while building a tree that expresses the distinct possibilities to be explored in the search of further MinAs. It first gets a arbitrary MinA S_0 for O, which is used to label the root of the tree. Then, for every axiom $a \in S_0$, a successor node is created and the axiom a is used to label the edge between root node and the successor node. If $O \setminus \{a\} \not\models \alpha$ then, this node is a dead node. In this case, we do backtracking by adding back the axiom a to O and removing the next axiom in S_0 from O.

If $O\backslash\{a\}$ still entails the consequence α, then we get another MinA S_1 from $O\backslash\{a\}$ using the black-box reasoner. This MinA is guaranteed to be distinct from S_0 since $a \notin S_1$. Then, for each axiom $a' \in S_1$, a new successor node is created and treated in the same way as the successors of the root node. That is, the algorithm checks whether $O \setminus \{a, a'\}$ still entails the consequence. This process terminates as an ontology is a finite set of axioms and the end result is a tree where every leaf node of the tree is a dead node. The path from root node to every dead node is a set of axioms, the minimal sets (w.r.t set inclusion) among these sets of axioms gives all the diagnoses $\mathcal{D}$ for $O \models \alpha$.

After all the diagnoses are obtained, for an arbitrary diagnosis $D_0 \in \mathcal{D}$, repair is computed as $R_0 = O \setminus D_0$. Thus, this algorithm computes all the diagnoses from the MinAs of a consequence and from the diagnoses, it computes the set of all the repairs for a consequence w.r.t an ontology. This HST-based approach to compute all the repairs for a consequence w.r.t a given ontology is explained using an example given below.

Example 2.6. *Consider the ontology shown in Figure 2.1 and the consequence $A \sqsubseteq F$. This example shows how to compute the repairs for $O \models A \sqsubseteq F$ using the HST-based method that is described above.*

Example 2.5 shows that, there exist two MinAs for the entailment, $O \models A \sqsubseteq F$. To construct the tree to compute diagnoses, we get an arbitrary MinA S_0 for $O \models A \sqsubseteq F$. Let S_0 be $\{a_1, a_5, a_6\}$. The axiom $a_1 \in S_0$ is removed from O. As $O \setminus \{a_1\} \not\models A \sqsubseteq F$ we get a dead node and the edge between root node and dead node is labelled with axiom "a_1". The HST based tree constructed is shown in Figure 2.4.

Then we backtrack by adding a_1 to O and removing next axiom from S_0, i.e., a_5. As $O \setminus \{a_5\} \models A \sqsubseteq F$, we add a successor node to the root node, labelled with an arbitrary MinA $S_1 = \{a_1, a_2, a_3, a_4, a_6\}$ (the MinA, S_1 is different from S_0, as $a_5 \notin S_1$). Then do the same process on S_1 as for the root node. We remove an axiom $a_1 \in S_1$ from O, at this point we removed two axioms from O: a_5 and a_1. As $O \setminus \{a_5, a_1\} \not\models A \sqsubseteq F$ we get a dead node.

In the similar manner, we build the complete tree as shown in Figure 2.4. In this tree, there are seven leaf nodes, which are dead nodes. Out of which five branches are minimal. Thus we obtain five diagnoses.

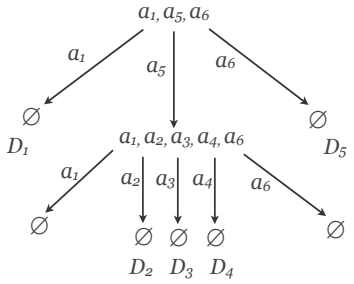

Figure 2.4: An expansion of the HST tree to compute diagnoses

The five diagnoses obtained from the tree contain the following axioms,
$D_1 = \{a_1\}$,
$D_2 = \{a_5, a_2\}$,
$D_3 = \{a_5, a_3\}$,
$D_4 = \{a_5, a_4\}$,
$D_5 = \{a_6\}$
From these diagnoses, repairs are computed as complement of the diagnosis as follows. R_1 *is computed as* $O \setminus D_1$. *In this manner we obtain 5 repairs,*
$R_1 = \{a_2, a_3, a_4, a_5, a_6\}$,
$R_2 = \{a_1, a_3, a_4, a_6\}$,
$R_3 = \{a_1, a_2, a_4, a_6\}$,
$R_4 = \{a_1, a_2, a_3, a_6\}$,
$R_5 = \{a_1, a_2, a_3, a_4, a_5\}$.
All these repairs are represented in Figure 2.2.

The HST-based procedure to compute all the repairs for a consequence w.r.t a given ontology is described in *Algorithm* 1. This algorithm takes an ontology O, a consequence α as input. It computes repairs using HST-based approach as described above and returns the set of repairs $\mathcal{R}$ for $O \models \alpha$ as output. The algorithm first computes an arbitrary MinA S using a the black-box reasoner in step 5. For every axiom a in S, a is removed from O and if $O \setminus \{a\} \models \alpha$ then, the algorithm is called recursively in step 10 to build the entire tree. Otherwise if $O \setminus \{a\} \not\models \alpha$ then, a dead node is reached in the tree and the path from every dead node to the root node is a set of axioms. All these sets of axioms are stored in the set $\mathcal{D}$ in the step 12. As we obtain a dead node, backtracking is done by adding back the axiom a to the ontology and the process continues until all the branches of the tree are explored. Then, all the non-maximal and duplicate elements are removed from $\mathcal{D}$ in the steps 15-19. After removing the non-maximal and duplicate elements from $\mathcal{R}$, it contains the set of all diagnoses for α w.r.t the ontology O. Then, all the repairs are computed from the diagnoses in steps 20-22. In this manner all the repairs are computed for a consequence using HST-based algorithm.

There exists many approaches to compute the repairs for a consequence w.r.t an

Algorithm 1 Computing repairs using Reiter's Hitting set tree algorithm

```
1: procedure ComputeRepairs(O, α)
2:     Input: O : Ontology, α : consequence
3:     Output: R: R is a set of repairs for O, α
4:     Global : D := ∅; Diagnosis, D := ∅; set of Diagnoses, S := ∅; S; MinA, R := ∅; a
   repair
5:     S ← MinA(O, α)                ▶ Gets a MinA for O, α from black-box reasoner
6:     for every axiom, a ∈ S do
7:         O ←O \ {a}
8:         D ←D ∪ {a}
9:         if O ⊨ α then
10:            call ComputeRepairs(O, α)
11:        else
12:            D ← D ∪ D
13:            D ←D \ {a}
14:            O ←O ∪ {a}
15:    for every D_i, D_j∈ D where (1≤i, j ≤| D |) do
16:        if D_i ⊆ D_j then
17:            D ← D \ {D_i}
18:        else if D_j ⊆ D_i then
19:            D ← D \ {D_j}
20:    for every D ∈ D do
21:        R ←O \ {D}                ▶ computes a repair from diagnosis
22:        R ← R ∪ R
23:    return(R)
24: end procedure
```

ontology. The following lemma shows the complexity to compute repairs for a consequence w.r.t an ontology.

Lemma 2.7. *It takes exponential time to compute repairs for a consequence w.r.t a given ontology.*

Proof: A repair is a sub-ontology of a given ontology O. As there exists exponentially many sub-ontologies in an ontology, it takes *exponential time* to compute all the sub-ontologies of O. After computing all the sub-ontologies, every sub-ontology can be assumed as a repair, check if it entails the consequence or not and if it is maximal. This can be done in polynomial time. Thus, we can conclude that, it takes *exponential time* to compute all the repairs for a consequence α w.r.t. an ontology O. □

Here we present a procedure to describe the repairs as contexts in the ontology. Every repair for a consequence w.r.t an ontology is considered as a context in the ontology. As all the repairs are incomparable and maximal, it ensures that all the contexts in the ontology are incomparable and maximal.

A new labelling function lab_c: $R \leftarrow N$ is used to assign label to every repair, where N represents natural numbers. The function lab_c maps a repair to a natural number, it assigns a unique label (a natural number) to every repair. Initially, if there are no contexts in the

ontology (that is, the ontology is not labelled), then the labelling of the contexts starts from the natural number "1". Whenever new repair is obtained, we check the highest value assigned to the existing contexts, let highest value (natural number) of the label assigned for the existing context be n. Then, for a new repair R_i to be described as context, the label, $lab_c(R_i) = n{+}1$ is assigned.

Every axiom that belongs to a context is annotated with the label of the context. Thus, every axiom is assigned with a label according to the contexts it belongs to. The ontology, thus obtained by annotating axioms with labels is called a labelled ontology (Definition 2.1). The step-by-step procedure to annotate the axioms of an arbitrary ontology O with labels, in which every context correspond to a repair in the set $\mathcal{R}$ w.r.t to an arbitrary axiom α is described below.

1. Let L be the set of labels in the ontology O. (If O is unlabelled, then $L{=}\emptyset$ and for every axiom $a \in O$, $lab(a) = \emptyset$)
2. Every repair in the set $\mathcal{R}$ is assigned with a unique label l ($l \in N$) such that $l \notin L$.
3. After assigning a label l to each repair, the set of labels L is updated as $L = L \cup l$.
4. For a repair $R_i \in \mathcal{R}$, let l_i be the label assigned to the repair R_i (i.e., $lab_c(R_i) = l_i$).
5. Every axiom $a \in R_i$, $lab(a) = lab(a) \cup l_i$.

In this manner, by annotating all the axioms of the ontology with labels, a labelled ontology is created. The procedure to label the axioms with context labels is explained in the example given below.

Example 2.8. *Consider the ontology O shown in Figure 2.1. Example 2.6 gives all the repairs for the consequence $A \sqsubseteq F$ w.r.t the ontology O. This example shows, how these repairs can be described as contexts in the ontology O.*
There are five repairs for $O \models A \sqsubseteq F$, these five repairs correspond to five contexts and every context is assigned with a label using the labelling function lab_c. As there are no contexts in the ontology O initially, the labels for the contexts start from the natural number "1".

All the repairs are enumerated with labels as follows.
$lab_c(R_1) = 1$,
$lab_c(R_2) = 2$,
$lab_c(R_3) = 3$,
$lab_c(R_4) = 4$,
$lab_c(R_5) = 5$.

After assigning labels to all the repairs, these labels are stored in the set, $L = \{1,2,3,4,5\}$. Using these labels, all the axioms $a \in O$ are annotated with labels. The labelling function $lab: O \rightarrow L$ assigns labels to every axiom of the ontology as follows.

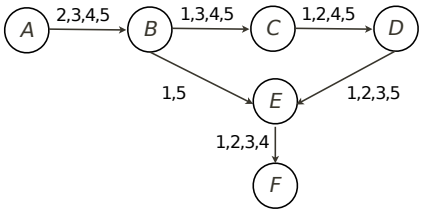

Figure 2.5: A labelled ontology

lab(a_1): {2, 3, 4, 5}, as axiom a_1 belongs to the repairs, R_2, R_3, R_4, R_5 this axiom is annotated with the labels corresponding to these repairs. In the same manner all the other axioms of O are assigned labels as follows.
lab(a_2): {1, 3, 4, 5},
lab(a_3): {1, 2, 4, 5},
lab(a_4): {1, 2, 3, 5},
lab(a_5): {1, 5},
lab(a_6): {1, 2, 3, 4}.

Thus we obtain the labelled ontology O by annotating all the axioms with labels. This labelled ontology is represented in Figure 2.5.

As shown in Example 2.6, there exists many repairs for a consequence w.r.t to an ontology. In the same manner, there exists many solutions for an ontology evolution task and it requires very high memory and computational effort to store all the solutions while doing ontology evolution iteratively. Due to this problem, in general, only one optimal solution which satisfies the principle of minimal change is chosen. To overcome this problem of storing every solution (which are partly redundant) separately, which forces to choose only one solution after each step, our approach offers to store all the solution compactly as a single ontology by annotating axioms of the ontology with labels. Example 2.8 shows how to compactly represent all the solutions by labelling axioms of the ontology which has the benefit of keeping only one ontology, instead of many solution and reasoning can be done easily over these labelled ontologies efficiently. The task of reasoning over a labelled ontology is called *Context-based reasoning*. The next section presents the procedure to do context-based reasoning.

2.1.2 Reasoning Over The Labelled Ontologies

As mentioned in the beginning of this chapter, the second step in context-based reasoning is to do reasoning over the labelled ontology. For a labelled ontology as introduced above, pre-computing that a certain consequence follows from the whole ontology is not sufficient. Pre-computing consequences for all the contexts is not a good idea since, then one might have to compute and store consequences for exponentially many different subsets of the

ontology. To overcome this problem, a *boundary* [10] is computed for a consequence which is defined below.

Definition 2.9 (boundary)**.** Given a labelled ontology O and a consequence α, let L be the set of labels in O. Then, the *boundary* of the consequence α is an element $\vartheta \subseteq L$ such that, a context, $O_l \models \alpha$ iff $l \in \vartheta$.

There exists many algorithms to compute the boundary for a consequence [10, 11]. One naive approach is to check every context of O, if the context entails the consequence α. If it entails, then, the label of the context is added to the boundary ϑ. If the context does not entail α then, the label of the context is not added to the boundary ϑ. Thus ϑ is computed by checking every context of O. Given below is an example to show how to compute the boundary.

Example 2.10. *Consider the labelled ontology O shown in Figure 2.5. The task is, to compute boundary for the consequence $B \sqsubseteq E$ w.r.t the ontology O.*

All the repairs represented in Figure 2.2 are described as contexts in the labelled ontology represented in Figure 2.5. The figure shows that $B \sqsubseteq E$ holds only the repairs R_1, R_5. The labels assigned to these repairs are, 1, 5.

Thus, boundary of $B \sqsubseteq E$ w.r.t labelled ontology O is $\vartheta = \{1, 5\}$.

The following lemma shows the complexity to compute the boundary for a consequence from a labelled ontology.

Lemma 2.11. *It takes exponential time to compute the boundary for a consequence α w.r.t a given ontology O.*

Proof: There exists exponential number of contexts in an ontology O. To check if a consequence α follows from a context of the ontology takes polynomial time. To check entailment from exponential number of contexts take *exponential time*. Thus, the complexity to compute boundary is *exponential time*. □

This section presented all the definitions of context-based reasoning method and also described the procedure to compute repairs for a consequence from the MinAs of the consequence w.r.t to an ontology using Hitting-set algorithm. It also described in detail, how to label the axioms of the ontology using the repairs as contexts in the ontology and the ways to do context-based reasoning over the labelled ontologies. The next section presents the definitions and basic details about the existing practical approaches to do ontology evolution operations.

2.2 Ontology Evolution Operations

In Belief Revision, the process of "incorporating" new information into a Knowledge Base (KB) maintaining the consistency of the KB is called *revision*. The process of "incorporating" new information into the KB without checking consistency of the KB is called

"update"and the process of "retracting" information that is no longer allowed to hold is called *contraction* [5]. All the ontology evolution operations should satisfy the *principle of minimal change* [5], according to which the semantics of a knowledge base should change "as little as possible", thus ensuring that modifications have least possible impact on the ontology. All the ontology evolution operations should satisfy the AGM [6] postulates. The evolution of an ontology O results in another ontology O' in which the required information is incorporated, retracted or updated.

2.2.1 Ontology Contraction

Ontology contraction operation is defined as follows.

Definition 2.12 (ontology contraction)**.** Given an arbitrary ontology O and an arbitrary consequence α, *ontology contraction* is an operation (denoted by O-α) to retract the consequence α from the ontology O. The result of the operation is the ontology O' which is a maximal subset of O such that O' has the following properties:

- $O \models O'$
- $O' \not\models \alpha$
- there exists no $O'' \subset O$ such that $O' \subset O''$ and $O'' \not\models \alpha$
- O' is as similar as possible to O ensuring minimal change.

The approaches adopted for ontology contraction in practice are *syntactic approaches* [7]. To retract an axiom entailed by an ontology, it suffices to compute all the MinAs for the given consequence w.r.t the given ontology. From the MinAs obtained, compute maximal subset of the ontology that does not entail the consequence (a repair) to be retracted. The solution obtained in this manner complies with the principle of minimal change. But, there exists many solutions (repairs). To ensure minimal change of the ontology, usually only the solution which removes minimum set of axioms from the ontology is chosen. This solution is called *optimal solution*. Given below is an example of ontology contraction.

Example 2.13. *Consider the ontology O shown in Figure 2.1 and consequence $\alpha = A \sqsubseteq F$, the example shows how to retract $A \sqsubseteq F$ from O.*

As $O \models A \sqsubseteq F$, to retract this consequence from O, it suffices to compute all MinAs for $A \sqsubseteq F$ w.r.t the ontology O. The MinAs for $A \sqsubseteq F$ w.r.t O are shown in Figure 2.3. The maximal subsets of O that does not entail the consequence (repairs) should be computed from the MinAs.

Example 2.6 shows how to compute the repairs from MinAs for a consequence. The repairs for $A \sqsubseteq F$ w.r.t the ontology O are shown in Figure 2.2.

All the repairs are possible solutions to retract $A \sqsubseteq F$ from O.
Among all the solutions (i.e., repairs) only two solutions satisfy the principle of minimal change. They are R_1 and R_5.
R_1 satisfies all the conditions of ontology contraction, $R_1 \subset O$, $R_1 \not\models A \sqsubseteq F$.
R_5 also satisfies the following conditions, $R_5 \subset O$, $R_5 \not\models A \sqsubseteq F$.
Both the repairs retract only one axiom from O and are optimal solutions for this contraction operation.

Ontology contraction operation should satisfy all the AGM [6] postulates for contraction.

2.2.2 Ontology Expansion

The next ontology evolution operation is ontology expansion. In ontology evolution the operations are performed on an ontology which is consistent. In ontology expansion operation, an axiom is added to the ontology. After adding the axiom, it is not required to check the consistency of the ontology. This operation is defined as follows.

Definition 2.14 (ontology expansion)**.** Given an arbitrary ontology O which is consistent and an axiom α, ontology expansion (denoted as $O+\alpha$) is the operation of adding the axiom α to the ontology O. The expanded ontology is O' should satisfy the following conditions:

- $O' = O \cup \alpha$
- $O' \models \alpha$

To add an axiom α to an ontology O, it suffices to check whether the ontology O entails α. If the ontology O does not entail α then, the axiom α is added to the ontology. Otherwise if $O \models \alpha$ then, the axiom α is not added to the ontology as the consequence α entails from the ontology. The following example demonstrates the ontology expansion operation.

Example 2.15. *Consider the ontology shown in Figure 2.1 and the axiom $\alpha = A \sqsubseteq G$, the task is to add $A \sqsubseteq G$ to the ontology O.*

As $O \not\models A \sqsubseteq G$, the axiom $A \sqsubseteq G$ should be added to the ontology O.
The updated ontology is $O' = O \cup \{A \sqsubseteq G\}$.
The ontology O' is shown in Figure 2.6.

The ontology is consistent after adding the axiom $A \sqsubseteq G$. Thus, the ontology O' is the result of the ontology expansion operation on O w.r.t the axiom $A \sqsubseteq G$.

Ontology expansion operation should satisfy all the AGM [6] postulates for expansion.

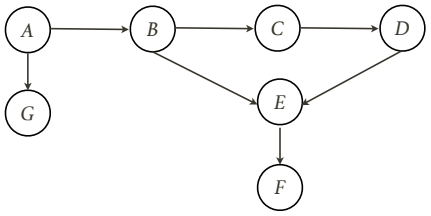

Figure 2.6: An ontology obtained as the result of ontology expansion operation

2.2.3 Ontology Revision

The next operation in ontology evolution is called ontology revision. This operation deals with the problem of inconsistency after adding an axiom to the ontology. That is, if the ontology becomes inconsistent after adding a new axiom then, the ontology revision operation retracts inconsistency from the ontology. The operation is defined as follows.

Definition 2.16 (ontology revision)**.** Given an arbitrary ontology O which is consistent and an axiom α, ontology revision (denoted by $O \circ \alpha$) is the operation of revising the axioms of ontology O to add the axiom α to O. The revised ontology O' should satisfy the following conditions:

- $O' \models \alpha$
- O' is consistent
- the symmetric difference between the set of axioms in O and the set of axioms in O' is minimal.

This operation consists of two sub-operations. The First step is ontology expansion on a given ontology and the given axiom. If the ontology becomes inconsistent after addition of the axiom then, the second step is to retract inconsistency from the updated ontology. Ontology revision operation should satisfy all the AGM [6] postulates for revision operation. The following example demonstrates the ontology revision operation.

Example 2.17. *Consider the ontology O shown in Figure 2.1 and the axiom $\alpha = A \sqsubseteq \neg E$ to be added to the ontology O. After adding the axiom, consistency of the ontology should be ensured.*

The first step is to add the consequence $A \sqsubseteq \neg E$ to the ontology O. As $O \not\models A \sqsubseteq \neg E$, it should be added to O. The ontology O' after adding $A \sqsubseteq \neg E$ is such that,
$O' = O \cup \{A \sqsubseteq \neg E\}$.
The ontology O' is shown in Figure 2.7. The ontology O' is inconsistent after adding the axiom. To retract inconsistency from the ontology O' by removing minimal set of axioms, ontology contraction operation should be done on the ontology O' w.r.t inconsistency.

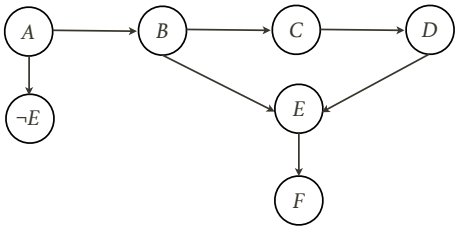

Figure 2.7: An ontology which becomes inconsistent after ontology expansion operation

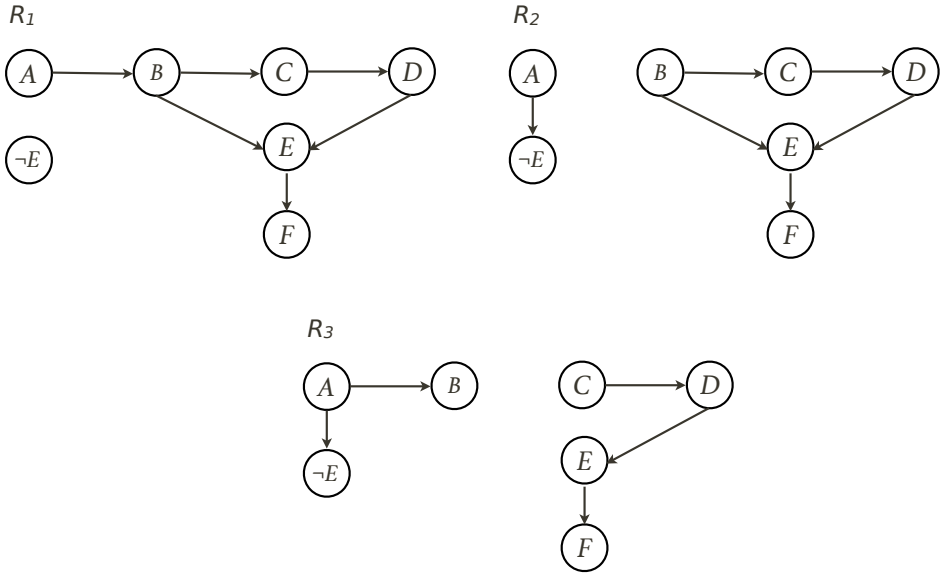

Figure 2.8: possible solutions for retracting inconsistency from the ontology

The possible solutions after retracting inconsistency are shown in Figure 2.8. There exists three solutions R_1, R_2, R_3.
R_1 cannot be chosen because, $R_1 \not\models A \sqsubseteq \neg E$, the axiom which should be added to the ontology.
Among the solutions R_2 and R_3, R_2 removes minimal set of axioms from O'.
Thus, R_2 is the optimal solution for ontology revision operation on the ontology O w.r.t the axiom $A \sqsubseteq \neg E$.

This concludes Chapter 2. This chapter presented details about context-based reasoning, ontology evolution operations and how context-based reasoning method can be used to do ontology evolution operations, which is very beneficial as it gives an advantage to store all the solutions during ontology evolution compactly, instead of choosing one optimal solution. We can now build on this background and see how ontology evolution operations can be done using context-based reasoning. This is elaborated in the next chapter in detail.

Chapter 3

Context-Based Ontology Evolution

This Chapter presents the theoretical methods to do ontology evolution operations: ontology contraction, ontology expansion and ontology revision using context-based reasoning method. As the ontology evolution operations are done using context-based reasoning, they are named as: context-based ontology contraction, context-based ontology expansion and context-based ontology revision respectively. This chapter presents the relationship between ontology evolution operations and context-based ontology evolution operations. It also describes theoretical methods to do context-based ontology evolution operations (contraction, expansion and revision). Section 3.1 describes a new approach to do context-based ontology contraction and presents an example to show how to do context-based ontology contraction operation. Section 3.2 presents a new approach to do context-based ontology expansion and explains the operation with an example. Section 3.3 describes about context-based ontology revision operation, which is a variant of context-based ontology expansion.

3.1 Context-Based Ontology Contraction

Contraction is the process of "retracting" information from a Knowledge Base, that is no longer allowed to hold [5]. Given an arbitrary ontology O and a consequence α, ontology contraction is the operation to remove the consequence α from the ontology O [5]. This operation is denoted as, $(O-\alpha)$. Ontology contraction operation is explained in detail in Chapter 2. Context-based ontology contraction is a new approach to do ontology contraction using context-based reasoning. It ensures that none of the contexts in the ontology entail the consequence.

Ontology contraction removes an axiom from the ontology. But, in context-based ontology contraction, the consequence is not retracted from the ontology, but axioms of the ontology are relabelled in such a way that, the consequence does not follow from any context of the relabelled ontology. To do context-based ontology contraction, we assume that the given ontology is labelled and for any arbitrary contexts O_i, O_j in an arbitrary labelled ontology O, $O_i \nsubseteq O_j$ and $O_j \nsubseteq O_i$. If the ontology is not labelled, then, the whole

ontology is considered as one context and all the axioms of the ontology are annotated with label "1". The operator "$-_c$" is called context-based ontology contraction operator, which signifies the context-based ontology contraction operation. This operation is defined as follows.

Definition 3.1 (context-based ontology contraction)**.** Given an arbitrary labelled ontology O and an arbitrary consequence α, let L be the set of labels in O. Then, *context-based ontology contraction* is a relabelling operation (denoted by $O'{=}O{-_c}\alpha$), which creates the labelled ontology O' with the relabelled axioms of O. Let L' be the set of labels in O' and for every label $l' \in L'$, the corresponding context is $O'_{l'}$, then every context $O'_{l'}$ in O' has the following properties.

- Every context $O'_{l'}$ in the ontology O' is a maximal sub-context of an arbitrary context O_l in O, (for a label, $l \in L$) such that $O'_{l'} \not\models \alpha$.
- For every context, $O'_{l'}$ in the ontology O' and an arbitrary context O_l in O, if $O'_{l'} \subset O_l$ then, every axiom $a \in \{\ O_l \setminus O'_{l'}\ \}$, $O'_{l'} \cup \{a\} \models \alpha$
- For every context O_l in O (for every $l \in L$), if there exists $S \subseteq O_l$, such that $S \not\models \alpha$, then $S \subseteq O'_{l'}$ (for an arbitrary label $l' \in L'$).
- $O' = O$

The theorem given below shows the relationship between ontology contraction and context-based ontology contraction.

Theorem 3.2. *Let O be an arbitrary labelled ontology and α be an arbitrary consequence, let O' be the labelled ontology such that $O' = O{-_c}\alpha$. And let O_B be an ontology which is the result of ontology contraction on O ($O_B = O{-}\alpha$). Then every context in the labelled ontology O' is a possible solution for ontology contraction on O w.r.t the consequence α.*

Proof. O_B is the result of ontology contraction on O w.r.t the consequence α. Then O_B has the following properties, $O_B{\subseteq}O$, $O_B{\not\models}\ \alpha$ and there exists no O'_B such that $O_B{\subset}O'_B$ and $O'_B{\not\models}\ \alpha$ [5].
The ontology O' is the result of context-based ontology contraction on the ontology O w.r.t the consequence α. Let L' be the set of labels in O' then, O' has the following properties, for every label $l' \in L'$, the corresponding context $O'_{l'}{\subseteq}O'$, $O'_{l'} \not\models \alpha$ and there exists no context, $O'_{l''}$ such that $O'_{l'}{\subset}O'_{l''}$ and $O'_{l''}{\not\models}\ \alpha$. As every context of the ontology O' satisfies the conditions of ontology contraction, every context is a possible solution for ontology contraction on the ontology O and O_B is one of the context in the labelled ontology O' as O_B is the optimal solution for ontology contraction on O w.r.t α. □

Consider a labelled ontology O which has the set of labels L and α be the consequence on which context-based ontology contraction operation should be done. A naive approach to do context-based ontology contraction is, to compute all the repairs for α from every context in the given ontology O (as a repair is a maximal sub-set of the context which does not entail the consequence α) and describe every repair as a context in the ontology O'. Thus, it ensures that every context in the labelled ontology O' do not entail α. This naive

approach is not efficient because it computes repairs for a given consequence from every context of the input ontology and relabels all axioms of the ontology.

To overcome this problem, we develop an algorithm to do context-based ontology contraction efficiently. In this algorithm, instead of computing repairs from every context in the ontology, boundary ϑ is computed for the consequence w.r.t the given labelled ontology. As $\vartheta \subseteq L$, which contains the labels of the contexts from which the consequence holds. All the contexts corresponding to the labels in the set $L \setminus \vartheta$ do not entail the consequence α. Thus, it is sufficient to compute repairs only from the contexts that correspond to the labels in ϑ and relabel only the axioms that belong to those contexts. All the axioms that belong to the contexts whose labels are present in the set $\{L \setminus \vartheta\}$ are not relabelled. The ontology thus relabelled ensures that the consequence is not entailed from any context of the relabelled ontology. This procedure is described step-by-step in Algorithm 2.

Algorithm 2 takes an arbitrary labelled ontology O, an arbitrary consequence α to be retracted from the ontology as input and returns a relabelled ontology O' such that none of the contexts in O' entail the consequence α. The algorithm declares the following global variables: $\mathcal{R}$ the set which is used to store all the repairs for α w.r.t every context corresponding to the labels present in the boundary for $O \models \alpha$, the variable S is used to represent the sub-ontology which contains all the axioms that belong to the contexts whose labels are present in the boundary. In step 6, the algorithm calls *computeBoundary* procedure to compute boundary ϑ for $O \models \alpha$.

After computing the boundary, it is sufficient to compute repairs for the consequence α w.r.t every context whose labels are present in ϑ. To do this task, Algorithm 2 creates a sub-ontology S with all the axioms in the contexts, whose labels are present in ϑ in step 10. The contexts whose labels are present in ϑ should be removed from O as they entail the consequence α. To remove those contexts from the ontology, it is sufficient to update the labels of the axioms in O such that, for every axiom $a \in O$, for every label $l \in \vartheta$, if $l \in$ *lab(a)* then, *lab(a)* is updated to *lab(a)* $\setminus\ l$, this is done in step 11. In step 13, the algorithm adds all the axioms whose labels are present in $L \setminus \vartheta$ to the ontology O' as the labels in $L \setminus \vartheta$ correspond to the contexts that does not entail the consequence α. Thus all those contexts are the repairs for α w.r.t O and they can be described as contexts in the labelled ontology O'.

In step 15, the algorithm calls the procedure *ComputeBoundaryRepairs* with parameters S and α. It returns all the repairs $\mathcal{R}$ to α w.r.t every context corresponding to the labels in ϑ. The repairs present in $\mathcal{R}$ may be contained in any of the contexts of O corresponding to the labels in $L \setminus \vartheta$ (as these contexts do not entail α), it is not required to describe such repairs as contexts in O' (as these contexts are already descried in O'). Thus, such repairs should be removed from $\mathcal{R}$. In step 19, Algorithm 2 removes such non-maximal elements from $\mathcal{R}$ w.r.t to the other contexts (which does not belong to the boundary) in O. Then all repairs R_j in $\mathcal{R}$ (where $1 \leq j \leq |\ \mathcal{R}\ |$) are enumerated with the labels and these labels are updated in the set of labels L. In step 25, all the axioms that belong to the repairs in $\mathcal{R}$ are added to the ontology O' with the updated labels. These labels are assigned to the axioms

Algorithm 2 Ontology Contraction using Context-based Reasoning

procedure CONTEXT-BASED-ONTOLOGYCONTRACTION(O, α)
  Input: O: labelled Ontology, α : *consequence*
  Output: O': the labelled ontology which is the result of $O -_c \alpha$
  Global: $\mathcal{R}$:=$\emptyset$, S:=$\emptyset$, O':=$\emptyset$
  $L \leftarrow$ set of labels in O
  $\vartheta \leftarrow$ *compute–boundary*(O, α)
  for every $a \in O$ **do**
    for every $l \in \vartheta$ **do**
      if $l \in$ *lab(a)* **then**
        $S \leftarrow S \cup \{a\}$
      else
        $O' \leftarrow O' \cup \{a\}$
    $L \leftarrow L \setminus \{l\}$
  $\mathcal{R} \leftarrow ComputeBoundaryRepairs(S, \alpha, \vartheta)$
  for every $l \in \vartheta$ **do**
    if $l \in$ *lab(a)* **then**
      lab(a) $\leftarrow$ *lab(a)* $\setminus \{l\}$
  for every $R_j \in \mathcal{R}$(where $1 \leq j \leq |\ \mathcal{R}\ |$) **do**
    for every $l \in L$ **do**
      if $R_j \subseteq O_l$ **then**
        $\mathcal{R} \leftarrow \mathcal{R} \setminus R_j$ ▶ removes the elements which are not maximal from $\mathcal{R}$
      else
        $l_{new} \leftarrow n$ ▶ $n \in N$ and $n \notin L$
        $L \leftarrow L \cup \{l_{new}\}$
        for every $a \in R_j$ **do**
          lab(a) $\leftarrow$ *lab(a)* $\cup \{l_{new}\}$
          $O' \leftarrow O' \cup \{a\}$
  return(O')
end procedure

such that if an axiom a is in any of the repair R_j then, *lab(a)* is updated with *lab(a)* $\cup\, l_{new}$ where l_{new} is the label assigned to the repair R_j. Thus, the ontology O' is created with the all the updated axioms of O. Thus Algorithm 2 returns the labelled ontology O' which has the property that, none of the contexts in O' entail the consequence α. The example given below shows how to do context-based ontology contraction.

Example 3.3. *Consider the labelled ontology O shown in Figure 3.1 and the axiom* $A \sqsubseteq F$ *to do context-based ontology contraction.*

Context-based ontology contraction operation on the labelled ontology O w.r.t $A \sqsubseteq F$ *is a three step process.*

Step 1: *Compute the boundary for* $A \sqsubseteq F$ *w.r.t the labelled ontology O.*

All the axioms in the ontology O are assigned with the label "1". Since there is only context in the ontology O, that is "O_1*" entails the consequence, the boundary for* $O \models A \sqsubseteq F$ *is,* $\vartheta = \{1\}$.

Step 2: *Compute repairs for* $A \sqsubseteq F$ *w.r.t* O_1.

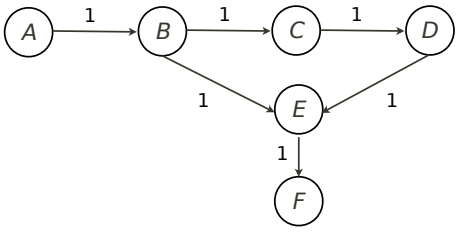

Figure 3.1: A labelled ontology

As all the axioms in O belong to the context O_1, repairs should be computed for $A \sqsubseteq F$ w.r.t all axioms of the ontology. Example 2.6 shows that, there exists five repairs for this entailment. As the label of the context O_1 is $1 \in \vartheta$ in the resulting ontology O', the label "1" is removed from the labels of all the axioms $a \in O_1$.

Step 3: *Describe all the five repairs computed as contexts in the ontology O'.*

Example 2.8 shows, how to describe the repairs as contexts in the ontology. There are two sub-processes involved in this step. They are,

(a)*Enumerate the repairs with labels:*

As the ontology O has one context, that is "1", the five repairs computed, that should be described as contexts in the labelled ontology O' get the labels from the natural number "2".

Thus, the five repairs gets the following labels.
$lab_c(R_1) = 2$;
$lab_c(R_2) = 3$;
$lab_c(R_3) = 4$;
$lab_c(R_4) = 5$;
$lab_c(R_5) = 6$;

(b) *label the axioms of O with context labels:*

For all the axioms of the context O_1, the old label (i.e., "1") is removed from the labels and new labels are assigned according to the contexts they belong to, as follows.
$lab(a_1) = (lab(a_1) \setminus \{1\}) \cup \{2,3,4,5\}$ *(as the axiom, a_1 belongs to the contexts 2, 3, 4 and 5). In this manner, all the axioms of the context, O_1 are relabelled.*

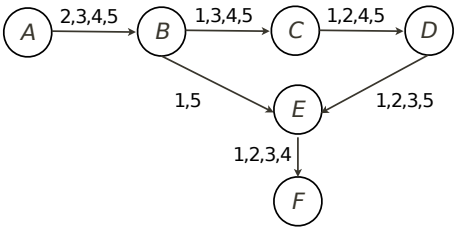

Figure 3.2: The labelled Ontology after removing the consequence

The new labels of the axioms are,
$lab(a_1)$: {*2,3,4,5*}
$lab(a_2)$: {*1,3,4,5*}
$lab(a_3)$: {*1,2,4,5*}
$lab(a_4)$: {*1,2,3,5*}
$lab(a_5)$: {*1,5*}
$lab(a_6)$: {*1,2,3,4*}
The labelled ontology, O' which contains the relabelled axioms is shown in the Figure 3.2.

There exists no context of the labelled ontology O' which entails the axiom $A \sqsubseteq F$.

Context-based ontology contraction is done by relabelling axioms of the ontology as shown in the example above. Thus, after context-based ontology contraction in the above example, there are five repairs but all the repairs are compactly represented as a single labelled ontology O' shown in Figure 3.2.

We presented the procedure to do context-based ontology contraction on an ontology w.r.t a consequence in Algorithm 2. There are three main steps in context-based ontology contraction, as shown in the example above. The second step in context-based ontology contraction is to compute repairs for the consequence w.r.t every context corresponding to the labels in the boundary. Here we present the step by step procedure to compute repairs for a consequence w.r.t every context corresponding to the labels in the boundary of an arbitrary consequence α w.r.t a labelled ontology O. Let ϑ be the boundary of α w.r.t O. The algorithm takes the boundary ϑ; the labelled ontology O_ϑ which contains all axioms corresponding to every context in ϑ; the consequence α as input. It returns set of repairs $\mathcal{R}$ which are set of all the repairs for α w.r.t every context $O_{\vartheta l}$ (for every label $l \in \vartheta$). The algorithm starts with computing repairs for α w.r.t the ontology O_ϑ.

In step 5, the algorithm calls the procedure *ComputeRepairs* which returns all the repairs $\mathcal{R}_{unprocessed}$ for α w.r.t input ontology O_ϑ. But, the repairs should be computed for α w.r.t every context $O_{\vartheta l}$ (for $l \in \vartheta$). Algorithm 3 does further processing of these repairs. For every repair, R_0 in $\mathcal{R}_{unprocessed}$, $R_0 \cap O_{\vartheta l}$ gives an set of axioms R_{0_l}, such that $R_{0_l} \not\models \alpha$ from the context $O_{\vartheta l}$, but it is not ensured that $R_{O_{\vartheta l}}$ is maximal w.r.t $O_{\vartheta l}$. In this manner, all the repairs in $\mathcal{R}_{unprocessed}$ are processed w.r.t every context in ϑ and these elements are

Algorithm 3 compute repairs for given consequence w.r.t to every context of the boundary

1: **procedure** ComputeBoundaryRepairs(O_ϑ, α, ϑ)
2: **Input:** O_ϑ: labelled ontology, α: consequence, ϑ: boundary of α
3: **Output:** $\mathcal{R}$: set of repairs for α w.r.t every context whose label is present in ϑ
4: **Global:** $\mathcal{R}:= \emptyset$; $\mathcal{R}_{unprocessed}:= \emptyset$; $\mathcal{R}_l:= \emptyset$; $R_{O_{\vartheta l}}:= \emptyset$;
5: $\mathcal{R}_{unprocessed} \leftarrow ComputeRepairs(O_\vartheta, \alpha)$ ▶ set of repairs for O, α
6: **for** every $R_0 \in \mathcal{R}_{unprocessed}$ **do**
7: **for** every $l \in \vartheta$ **do**
8: $R_{O_{\vartheta l}} \leftarrow R_0 \cap O_{\vartheta l}$ ▶ a repair for the context corresponding to the label l
9: $\mathcal{R}_l \leftarrow \mathcal{R}_l \cup \{R_{O_{\vartheta l}}\}$ ▶ set of repairs for the context corresponding $O_{\vartheta l}$
10: **for** every R_i, $R_j \in \mathcal{R}_l$ (where $1 \leq i < j \leq |\ \mathcal{R}_l\ |$) **do**
11: **if** $R_i \subseteq R_j$ **then**
12: $\mathcal{R}_l \leftarrow \mathcal{R}_l \setminus \{R_i\}$
13: **else if** $R_j \subseteq R_i$ **then**
14: $\mathcal{R}_l \leftarrow \mathcal{R}_l \setminus \{R_j\}$
15: $\mathcal{R} \leftarrow \mathcal{R} \cup \{\mathcal{R}_l\}$
16: **for** every R_i, $R_j \in \mathcal{R}$ (where $1 \leq i < j \leq |\ \mathcal{R}\ |$) **do**
17: **if** $R_i \subseteq R_j$ **then**
18: $\mathcal{R} \leftarrow \mathcal{R} \setminus \{R_i\}$
19: **else if** $R_j \subseteq R_i$ **then**
20: $\mathcal{R} \leftarrow \mathcal{R} \setminus \{R_j\}$
21: return($\mathcal{R}$)
22: **end procedure**

present in $\mathcal{R}_l$, then, the algorithm removes duplicate and non maximal elements from $\mathcal{R}_l$ in step 10. After removing such elements from $\mathcal{R}_l$ all the remaining elements in $\mathcal{R}_l$ are added to the set $\mathcal{R}$.

But, there might exist some duplicates or non maximal elements in $\mathcal{R}$. Such elements are removed from $\mathcal{R}$ in the steps 16-20. After removing such elements, the set $\mathcal{R}$ contains all the repairs for the consequence α w.r.t every context corresponding to the labels in ϑ. Thus, the algorithm returns the set $\mathcal{R}$ which contains set of all the repairs for α w.r.t every context $O_{\vartheta l}$ (for $l \in \vartheta$) and none of the repairs are contained in each other.

Theorem 3.4 given below proves the correctness of Algorithm 3.

Theorem 3.4. *Given a labelled ontology O, a consequence α and the boundary of α w.r.t O is ϑ, Algorithm 3 returns the set of all the repairs for α w.r.t every context whose labels are present in ϑ and none of the repairs are contained in each other.*

Proof : Algorithm 3 returns the set of elements $\mathcal{R}$, we should prove the following.
1. For some arbitrary element $R \in \mathcal{R}$, R is a repair for α w.r.t to an arbitrary context O_l, where $l \in \vartheta$. For R to be a repair for α w.r.t to an arbitrary context O_l, where $l \in \vartheta$, it should satisfy two conditions,
(a) $R \not\models \alpha$: We know that, for an arbitrary set of axioms, R_i and for some set $S \subseteq R_i$, if $R_i \not\models \alpha$ then, $S \not\models \alpha$. In Algorithm 3, the set of repairs, $\mathcal{R}_{unprocessed}$ are given by Algorithm

1. We know that, every element $R_0 \in \mathcal{R}_{unprocessed}$ is a repair for α w.r.t the ontology O. Every element, R in $\mathcal{R}$ is computed as, $R = R_0 \cap O_l$, where O_l is a context in O (for $l \in \vartheta$) which means $R \subseteq R_0$. It implies that, $R \not\models \alpha$ w.r.t the context O_l from which it is computed.
(b) R is maximal w.r.t the context from which it is computed: For R to be maximal w.r.t a context, O_l, there exists no $R' \in \mathcal{R}$ (w.r.t O_l) such that, $R' \subset R$ and $R' \not\models \alpha$. If there exist such R' in O_l then, Algorithm 3 removes such elements, R', which are duplicate and not maximal from $\mathcal{R}$. Thus, if an element, R is in $\mathcal{R}$ then, it is maximal w.r.t the context from which R is computed.

Since $R \not\models \alpha$ and R is maximal w.r.t the context from which it is computed, we can conclude that R is a repair for α w.r.t the context O_l from which it is computed.

After computing all repairs w.r.t each context in ϑ, as the algorithm ensures to remove all duplicate and not maximal elements from $\mathcal{R}$, we can also conclude that, for any repairs R_1, $R_2 \in \mathcal{R}$, it holds that $R_1 \not\subseteq R_2$ and $R_2 \not\subseteq R_1$.
2. If R_i is an arbitrary repair for α w.r.t a context O_l, where $l \in \vartheta$, then, $R_i \in \mathcal{R}$. If R_i is an arbitrary repair, then $R_i \not\models \alpha$ and R_i should be maximal w.r.t the context from which R_i is computed. As proved for the previous case that, the algorithm computes all the repairs in $\mathcal{R}$ such that, all the repairs do not entail α and are maximal. As R_i is a repair (it is maximal and does not entail α), it should be in $\mathcal{R}$. If R_i is not in $\mathcal{R}$ then, R_i is not a repair. It contradicts our assumption that, R_i is a repair. Thus, we can conclude that $R_i \in \mathcal{R}$. □

Theorem 3.5 proves the correctness of Algorithm 2 which is used to do context-based ontology contraction on a consequence w.r.t a given labelled ontology.

Theorem 3.5. *Given a labelled ontology O and a consequence α as input to Algorithm 2, then it returns the labelled ontology O' such that every context in O' is a repair for the consequence α w.r.t an arbitrary context in the input labelled ontology O and all the repairs for α w.r.t every context in the labelled ontology O are contained in the contexts of the labelled ontology, O'.*

Proof : Let L be the set of labels in the input labelled ontology O and L' be the set of labels in the labelled ontology O'. we should prove the following.
1. For every label $l' \in L'$, the corresponding context $O'_{l'}$ in the labelled ontology O', is a repair for the consequence α w.r.t an arbitrary context in the ontology O. There exists two cases for the context $O'_{l'}$.
(a) If the label $l' \in L$, it implies that $l' \notin \vartheta$ as, $O_{l'} \not\models \alpha$. Thus, this context is unchanged by the algorithm and it remains as a context in the labelled ontology O'. That is, the context $O'_{l'}$ is equivalent to $O_{l'}$. As $O_{l'} \not\models \alpha$ and there exists no context, $O_{l''}$ in O such that $O_{l'} \subset O_{l''}$. Thus, the context $O'_{l'}$ is repair for the consequence α w.r.t the context $O_{l'}$ in O, as, $O'_{l'} \not\models \alpha$ and $O'_{l'}$ is maximal.
(b) If the label, $l' \notin L$ then, the context, $O'_{l'}$ does not exist in O. It is a context in O', created by computing repairs to α w.r.t an arbitrary context O_l in O. As $O_l \models \alpha$, the

label, $l \in \vartheta$ and there exists a set of repairs $\mathcal{R}_l$ for α w.r.t the context O_l. We should prove that, there exists a repair, $R_{l_i} \in \mathcal{R}_l$ such that, $R_{l_i} \subseteq O'_{l'}$. We proved in Theorem 3.4 that, Algorithm 3 computes set of all repairs for a consequence w.r.t to a given context, as Algorithm 2 to do context-based ontology contraction, calls Algorithm 3 to compute repairs for the consequence α w.r.t the context O_l. It returns the set of repairs, $\mathcal{R}_l$ to α w.r.t the context O_l. And Algorithm 2 uses these repairs to describe contexts in the ontology O'.

If the repair, R_{l_i} is contained in an arbitrary context $O'_{l'}$ in O', then, a new context is not created for R_{l_i}, as $R_{l_i} \subset O'_{l'}$. If R_{l_i} is not contained in any context of the labelled ontology O', then, a new context $O'_{l'}$ is created in the labelled ontology O'. If R_{l_i} is not contained in a context in O', then it contradicts our assumption that R_{l_i} is a repair for α w.r.t to the context O_l in O. But it is proved that, R_{l_i} is a repair, thus we conclude that R_{l_i} is contained in a context in the ontology O'.
2. Let R_i be an arbitrary repair for α w.r.t an arbitrary context O_l in the ontology O, then, it should be proved that, the repair R_i is contained in an arbitrary context $O'_{l'}$ in O'. For the repair R_i there exists two cases.
(a) If $O_l \not\models \alpha$, it means that, the repair, $R_i = O_l$. That is, the label corresponding to O_l, $l \notin \vartheta$. Thus, the context O_l remains unchanged in the output ontology O' of Algorithm 2. The context in O' corresponding to the context O_l is O'_l. Thus, we conclude that, the repair R_i for α w.r.t a context O_l is represented as a context in the labelled ontology O'.
(b) If $O_l \models \alpha$, it means that, the repair $R_i \subset O_l$. In cases where, a context in O are such that, $O_l \models \alpha$, it means, the label, $l \in \vartheta$. Then, Algorithm 2 calls the procedure described in Algorithm 3 to compute all the repairs for α w.r.t the context O_l. As we proved in Theorem 3.4 that, Algorithm 3 computes set of all repairs $\mathcal{R}$, for α w.r.t O_l, R_i should be in $\mathcal{R}$. In case 1 (b) of this theorem, we proved that, all the repairs in $\mathcal{R}$ are contained in the contexts in the labelled ontology O', thus, R_i should be contained in a context in O'. If R_i is not contained in a context in O', it contradicts our assumption that R_i is repair w.r.t the context O_l. Thus, we conclude that, the repair R_i is contained in a context in O'. □

The lemma given below shows the complexity of Algorithm 2 to do context-based ontology contraction operation.

Lemma 3.6. *Given an arbitrary labelled ontology O and a consequence α to do context-based ontology contraction then, Algorithm 2 returns the labelled ontology O' as the result of context-based ontology contraction on O w.r.t the consequence α in exponential time.*

Proof : The complexity of Algorithm 2 depends on the complexity to compute boundary for the entailment $O \models \alpha$ and the complexity to compute repairs over the axioms of the contexts that belong to the boundary. From lemma 2.11, we know that, the complexity to compute boundary is *exponential time*. After the boundary is obtained, it takes *exponential time* to compute repairs for the consequence, α w.r.t every context, whose labels are present in the boundary (from lemma 2.7). And it takes polynomial time to describe the repairs computed, as contexts in the resulting labelled ontology O'. Thus, the overall running time of Algorithm 2 is *exponential time*. □

This section discussed about context-based ontology contraction, it presented the theoretical methods developed to do context-based ontology contraction operation. The next section elaborates context-based ontology expansion operation and describes a new approach to do context-based ontology expansion.

3.2 Context-Based Ontology Expansion

Given an ontology O, which is consistent and an axiom α, ontology expansion is the operation of adding α to the ontology O. This operation is denoted using the notation $(O{+}\alpha)$. Context-based ontology expansion is the operation of doing ontology expansion using context-based reasoning. The operator "$+_c$" is used to denote the operation context-based ontology expansion.

As in context-based ontology contraction, in context-based ontology expansion also the given ontology should be labelled. If the ontology is not labelled, all the axioms of the ontology are labelled with the label "1". In ontology expansion operation, an arbitrary axiom α is added to a consistent ontology. But, in context-based ontology expansion, the consequence α is added to every context of the labelled ontology in which every context is consistent. This operation is defined as follows.

Definition 3.7 (context-based ontology expansion)**.** Given an arbitrary labelled ontology O and an arbitrary axiom α, *context-based ontology expansion* is a relabelling operation which creates the labelled ontology O' from the given ontology O ($O'{=}O{+_c}\alpha$), by relabelling (or updating the label of) the axiom α. Let L' be the set of labels in the labelled ontology O' then, O' has the following properties.

- For every label $l \in L'$ the corresponding context O'_l in O' is such that $O'_l \models \alpha$.
- $L' = L$. That is, the set of labels, L' in O' is same as set of labels L in O.
- For every label, $l \in L$ the corresponding context $O_l \subseteq O'_l$, where O'_l is the context corresponding to the label l in the ontology O'.

The resulting labelled ontology O' has the following property.

- $O' = O \cup \alpha$

In order to relabel the axiom α, we should consider the following cases. Let L be the set of labels in the ontology O and let ϑ be the boundary of α w.r.t to the ontology O. If $O \not\models \alpha$, then we add the axiom α to every context of O i.e., the axiom α should be added to the ontology with the label, $lab(\alpha){=}L$. If, only few contexts of O entail α then, the boundary of α w.r.t O gives the labels of the contexts that entail α. It means that, we need to add α to the contexts corresponding to the labels in $L \backslash \vartheta$. The next case is, if the boundary of α w.r.t O is empty, then, it means that, $O \models \alpha$ but not from any context of O. Thus, the label of α should be updated to L. The scenarios described above are shown in Figure 3.3.

Entailment Status of *(O, ∝)*	**Boundary of** *(O, ∝)*	**Label of** ∝
Yes	B≠∅	L \ B
Yes	B=∅	L
No	B=∅	L

Figure 3.3 The table shows the label to be assigned to a newly added axiom in context-based ontology expansion operation

Rank

In our work, for context-based ontology expansion and revision, we consider cases, where not all axioms in an ontology have the same status, but some are preferred over the others. We model this preference by a simple *rank function*, *rank(a)*: $O \leftarrow N$ where N denotes Natural numbers greater than 0 and a is an axiom in the given ontology. The rank function maps every axiom of the ontology O, which is given as input to context-based ontology expansion or revision operations to the natural number "1".

The axioms with rank "1" are the highest ranked axioms. If the axiom is newly added during context-based ontology expansion or revision then, rank of that axiom will be mapped to a natural number greater than "1". The rank assigned to the newly added axiom depends on the highest natural number assigned to the rank of the existing axioms in the ontology. Suppose, if the highest natural number assigned to the rank of existing axioms in the ontology is n, then, rank for the newly added axiom during context-based ontology expansion or revision will be *n+1*.

The axioms of the ontology with rank "1" are the highest ranked axioms of the ontology. The rank of the axioms decreases as the value of the natural number assigned to the rank of the axiom increases. The rank function is required when a context needs to be extracted from the ontology which is very similar to the original ontology then, the context with highest rank can be extracted. The theorem given below, shows the relationship between ontology expansion and context-based ontology expansion operations.

Theorem 3.8. *Let O be an arbitrary labelled ontology in which every context is consistent and α be an arbitrary axiom, let O' be the labelled ontology such that $O' = O+_c\alpha$. Let O_E be the ontology such that $O_E = O+\alpha$. Then, every context in the labelled ontology O' is a possible solution for ontology expansion on O w.r.t the axiom α.*

Proof. O_E is the result of the ontology expansion operation on O w.r.t the axiom α. Then O_E has the following properties. $O_E = O \cup \alpha$, $O_E \models \alpha$.
The ontology O' is the result of context-based ontology expansion operation. Let L' be the set of labels in O'. Then O' has the following properties, for every context O'_l in O', $O'_l \subseteq O'$,

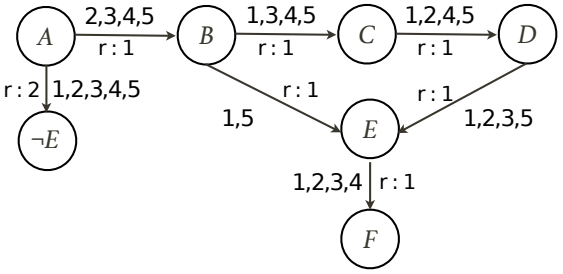

Figure 3.4: A labelled ontology after adding a consequence

$O'_l \models \alpha$. Thus, every context in the labelled ontology O' is a possible solution for ontology expansion operation on O w.r.t the axiom α. □

Given below is an example to demonstrate how context-based ontology expansion is done.

Example 3.9. *Consider the labelled ontology O shown in Figure 3.2 and the axiom* $A \sqsubseteq \neg E$ *to do context-based ontology expansion on the ontology O.*
This labelled ontology has five contexts i.e., the set of labels in O are,
$L = \{1, 2, 3, 4, 5\}$.
As this ontology does not contain ranks, ranks are added to all the axioms of the ontology. Initially, every axiom of the ontology is assigned with rank "1". As described above, to add the axiom $A \sqsubseteq \neg E$ *to the ontology O, first we should compute boundary for* $A \sqsubseteq \neg E$ *w.r.t O.*

As $O \not\models A \sqsubseteq \neg E$*, the boundary is,*
$\vartheta = \emptyset$.
It means that, $A \sqsubseteq \neg E$ *should be added to every context of the ontology O. That is,* $lab(A \sqsubseteq \neg E) = \{1, 2, 3, 4, 5\}$ *is assigned to this axiom and then, the axiom is added to O.*

As the highest natural number assigned to the ranks of existing axioms of the ontology is "1", we assign rank as "2" to the new axiom. The ontology O after adding $A \sqsubseteq \neg E$ *is shown in Figure 3.4.*

Here we describe a new approach to do context-based ontology expansion operation. The step by step procedure to do context-based ontology expansion for a given input labelled ontology O and an axiom α is described in Algorithm 4. The algorithm takes a labelled ontology O and an axiom α as input. It returns the labelled ontology O' such that α holds from every context of O'. The algorithm uses the following global variables: O' is the labelled ontology which is given as output by the algorithm, ϑ stores the boundary of α w.r.t O, $lab(\alpha)$ stores the set of labels that should be assigned to α, L stores the set of labels in O and L' stores the set of labels in O'. The first step in the algorithm is checking the entailment of α w.r.t O. If $O \models \alpha$ then, the algorithm computes boundary of the entailment. Otherwise, if $O \not\models \alpha$ or, if the boundary of $O \models \alpha$ is $\emptyset$ then, it implies the

Algorithm 4 Add axiom to all contexts of the ontology

1: **procedure** AddAxiomToAllContextsOfOntology(O, α)
2: **Input:** O: labelled ontology, α: the consequence to be added to every context of the output labelled ontology O'
3: **Output:** O': labelled ontology such that α holds from every context of O'
4: **Global:** O':=$\emptyset$, $\vartheta := \emptyset$, *lab*(α):= $\emptyset$, L: set of labels in the ontology O
5: **if** $O \models \alpha$ **then**
6: $\vartheta \leftarrow$ *compute–boundary*(O, α)
7: **else**
8: **for** every $l \in L$ **do**
9: *lab*(α) $\leftarrow$ *lab*(α) $\cup \{l\}$
10: **if** $\vartheta \neq \emptyset$ and $\vartheta \neq L$ **then**
11: **for** every $l \in (L \setminus \vartheta)$ **do**
12: *lab*(α) $\leftarrow$ *lab*(α) $\cup \{l\}$
13: **else**
14: **for** every $l \in L$ **do**
15: *lab*(α) $\leftarrow$ *lab*(α) $\cup \{l\}$
16: **for** every axiom, $a \in O$ **do**
17: $O' \leftarrow O' \cup \{a\}$
18: $O' \leftarrow O' \cup \{\alpha\}$
19: return(O')
20: **end procedure**

ontology O does not entail the consequence or α does not follow from any context of the ontology O. In any of these two cases, α should be added to all the contexts of O i.e., label assigned to α is the set of all labels of O this is done in the steps 9 and 15 and the axiom α is added to the ontology O. If the boundary ϑ is not empty then, it implies that α follows from the contexts of O whose labels are present in ϑ and α should be added to contexts of O corresponding to the labels $L\backslash\vartheta$. This process is done in step 12 of the algorithm. After assigning the label to α, the algorithm adds α to O in step 16. In step 18, the algorithm adds all axioms of O to the ontology O' and returns the labelled ontology O' in which every context entails α.

Theorem 3.10. *Given a labelled ontology O in which every context of the ontology is consistent and a consequence α, Algorithm 4 returns the labelled ontology O' such that every context in O' entails the consequence α.*

Proof : This is a special case of theorem 3.15. □

Lemma 3.11. *Algorithm 4 returns a labelled ontology O' in* exponential time*, such that every context of O' entails a given consequence α.*

Proof : This is a special case of lemma 3.16. □

This section discussed in detail about context-based ontology expansion and showed a new approach to do context-based ontology expansion. The next section shows a variant of context-based ontology expansion, that is context-based ontology revision operation. The difference between the two operations is that, in context-based ontology expansion, after

adding an axiom to the ontology, it is not required to ensure the consistency of the contexts of the ontology. In context-based ontology revision operation, after adding the axiom to the ontology, it is required to ensure that, every context of the ontology is consistent and every context entails the axiom that is added to the ontology in context-based ontology revision operation.

3.3 Context-Based Ontology Revision

Given an arbitrary ontology O which is consistent and an axiom α, ontology revision is the operation which revises the axioms of O to add the axiom α to O ensuring consistency of O [15], it is denoted as $O \circ \alpha$. Ontology revision is discussed in detail in chapter 1. Context-based ontology revision is a new approach to do ontology revision using context-based reasoning. The symbol "$\circ_c$" is used to denote the operation of context-based ontology revision.

As in context-based ontology contraction and expansion, in context-based ontology revision also the input ontology should be labelled. If the ontology is not labelled, all the axioms of the ontology are labelled with "1". In ontology revision operation, an arbitrary axiom α is added to the ontology and then consistency of the ontology is ensured. But, in context-based ontology revision, the axiom α is added to every context of the labelled ontology and consistency of every context of the ontology should be ensured after adding α. This operation is defined as follows.

Definition 3.12 (context-based ontology revision)**.** Given an arbitrary labelled ontology O and an arbitrary axiom α, *context-based ontology revision* ($O \circ_c \alpha$) is a relabelling operation, which creates the labelled ontology O' from the given ontology O, by relabelling the axioms of O. Let L' be the set of labels in O'. The labelled ontology O' has the following properties,

- for every label, $l' \in L'$, the corresponding context $O'_{l'} \models \alpha$.
- for every label, $l' \in L'$, the corresponding context $O'_{l'}$ is consistent.
- for every label, $l' \in L'$, the corresponding context $O'_{l'} \subseteq O_l \cup \alpha$, where O_l is an arbitrary context in the ontology O and l is a label for the context O_l.

After adding the axiom to every context of the labelled ontology, the consistency of every context of O should be ensured. To do that, we check consistency of every context, if any of the contexts are inconsistent then, inconsistency should be retracted from those contexts. In context-based ontology revision process, to get rid of inconsistency from the contexts, we compute the boundary of inconsistency, the boundary for inconsistency gives the labels of the contexts which are inconsistent. Then, repairs for these contexts are computed, to retract inconsistency. After all the repairs are computed, we are interested only in those repairs in the ontology O' which entail the consequence α. The following theorem shows the relationship between ontology revision and context-based ontology revision operations.

Theorem 3.13. *Let O be an arbitrary labelled ontology and α be an arbitrary axiom, let O' be the labelled ontology such that $O' = O \circ_c \alpha$. And let O_R be an ontology such that $O_R = O \circ \alpha$. Then every context in the labelled ontology O' is a possible solution for ontology revision operation on O w.r.t the axiom α.*

Proof. O_R is the result of ontology revision on O w.r.t the axiom α. Then O_R has the following properties. O_R is consistent, $O_R \models \alpha$ [15].
The ontology O' is the result of context-based ontology revision on O w.r.t the axiom α. Let L' be the set of labels in O'. Then O' has the following properties, every context O'_l in O', O'_l is consistent, $O'_l \models \alpha$. Thus, every context in the labelled ontology O' is a possible solution for ontology revision on O w.r.t the axiom α. □

The following example shows the difference between context-based ontology expansion and context-based ontology revision operations. It shows every step of context-based ontology revision in detail and how inconsistency is retracted from the contexts of the ontology.

Example 3.14. *Consider the labelled ontology O shown in Figure 3.2 and the axiom $A \sqsubseteq \neg E$ to do context-based ontology revision.*

There are two steps in context-based ontology revision operation on O. They are, context-based ontology expansion and second step is to remove inconsistency from every context of the ontology. The first step, context-based ontology expansion is shown in Example 3.9. The labelled ontology O'' after the first step is shown in Figure 3.4.

This labelled ontology shown in the figure has five contexts i.e., the set of labels in O'' are,
L''={1, 2, 3, 4, 5}. Out of the five contexts, the context, O_5 is inconsistent after adding the consequence $A \sqsubseteq \neg E$.
Thus, in the second step, inconsistency should be retracted from this context. To retract inconsistency from the context O_5, we compute repairs for inconsistency w.r.t O_5.

There are three repairs for inconsistency in the context O_5. They are,
R_1={$A \sqsubseteq \neg E$, $B \sqsubseteq E$},
R_2={$A \sqsubseteq B$, $B \sqsubseteq E$},
R_3={$A \sqsubseteq \neg E$, $A \sqsubseteq B$}.
The repairs R_1,R_2 and R_3 are shown in Figure 2.7.

But, all the three repairs cannot be added to the ontology O'' as the repair, $R_1 \not\models A \sqsubseteq \neg E$. We should add only those repairs to O', which entail the consequence $A \sqsubseteq \neg E$. Thus, only the repairs R_2 and R_3 can be added to the ontology O'' as contexts. The ontology after adding the repairs R_2 and R_3 is the labelled ontology O'. The ontology O' is shown in Figure 3.5.

Here we describe a step by step procedure to do context-based ontology revision for a given input labelled ontology O and an axiom α to be added to the ontology. This procedure

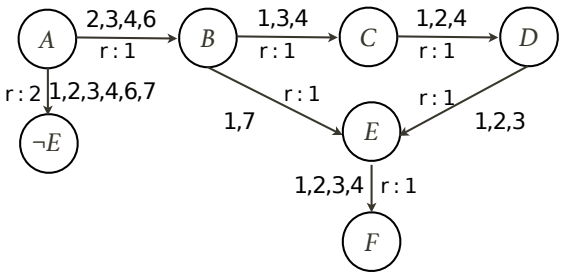

Figure 3.5 A labelled ontology obtained as the result of context-based ontology revision

Algorithm 5 context-based ontology revision

1: **procedure** CONTEXT-BASEDONTOLOGYREVISION(O, α)
2: **Input:** O: labelled ontology, α: the consequence to be added to every context of the output labelled ontology O'
3: **Output:** O': the labelled ontology which is the result of $O \circ_c \alpha$ operation
4: **Global:** $O'=\emptyset$; labelled ontology
5: $O'' \leftarrow$ AddAxiomToAllContextsOfOntology(O,α)
6: $O' \leftarrow$ RemoveInconsistencyFromContexts(O'', $\top \sqsubseteq \bot$)
7: return(O')
8: **end procedure**

is described in Algorithm 5. The algorithm takes a labelled ontology O and an axiom α as input. It returns the labelled ontology O' such that α holds from every context of O' and every context of O' is consistent. The algorithm declares the following global variables: O'' is the labelled ontology in which every context entails the consequence α, O' is the labelled ontology that is given as output by the algorithm in which every context entails α and every context is consistent. The algorithm first calls the procedure in Algorithm 4 which returns the labelled ontology O'' which entails the consequence α from every context. In the next step, Algorithm 5 calls the procedure in Algorithm 6 with O'' and α as input parameters. It returns the ontology O' in which every context is consistent and entails the consequence α. Thus Algorithm 5 gives the labelled ontology O' as the output of context-based ontology revision operation.

The two steps in context-based ontology revision: adding the axiom to every context of the ontology and ensuring consistency of every context of the ontology are described as two algorithms, Algorithm 4 and Algorithm 6. Algorithm 4 is described in detail in the previous section. Now we describe step by step procedure of Algorithm 6.

The second step of context-based ontology revision is retracting inconsistency from the contexts of the ontology O'' which is the output of Algorithm 4 and ensure that, after removing inconsistency every context of the ontology entails the consequence α. This procedure is described in Algorithm 6. The algorithm takes the ontology O and the consequence α as input. It returns the labelled ontology O' as output. The algorithm uses the following global variables: O' is the labelled ontology given as output, L is set of labels in the input ontology, L' is set of labels in the output ontology, $\vartheta_\bot$ stores the boundary of $\bot$ w.r.t the input ontology, $O_{\vartheta_\bot}$ stores all the axioms of the contexts whose labels are present

in $\vartheta_\perp$, $\mathcal{R}_{\vartheta_\perp}$ stores all the repairs for $\perp$ .w.r.t every context whose labels are present in $\vartheta_\perp$. The algorithm first computes boundary for inconsistency to find out which contexts of O are inconsistent after adding α in the step 5. The algorithm adds all the contexts of O that are consistent, to the labelled ontology O' in step 16. In step 18, it computes repairs for inconsistency by calling the procedure in Algorithm 3 which returns set of all repairs for inconsistency w.r.t every context in the ontology $O_{\vartheta_\perp}$.

After computing the repairs, Algorithm 6 removes those repairs from $\mathcal{R}_{\vartheta_\perp}$, which are contained in the existing contexts in the ontology O' in step 22. Then, in step 26, the algorithm checks, if every repair entails the consequence α. A repair is added as a context to the ontology O' iff it entails α, this is done in step 31. In this manner, we update labels of the axioms in O and create the new labelled ontology O' such that α follows from every context of O' and every context of O' is consistent.

Theorem 3.15. *Given a labelled ontology O in which every context of the ontology is consistent and a consequence α, Algorithm 5 returns the labelled ontology O' such that, every context in O' is consistent and every context entails the consequence α.*

Proof : Let L be the set of labels in the labelled ontology O and L' be the set of labels in O'. For every label $l' \in L'$, we should prove the following.
1. The context, $O'_{l'}$ in the labelled ontology O' is consistent. Let ϑ be the boundary of α w.r.t the input ontology O. Depending on the labels present in ϑ, there exists two cases,
(a) If $(\vartheta = L)$ then, it implies that α holds from all the contexts of O and it is not required to add α to any context of O. In this case, the set of labels $L' = L$. Thus, the contexts in O' are equivalent to the contexts in O. As every context in the input labelled ontology O is consistent, every context in the labelled ontology O' is also consistent.
(b) If $(\vartheta \subset L)$ then, it implies that α holds from only few contexts of O corresponding to the labels, that belong to ϑ. If the label $l' \in \vartheta$, as all the contexts corresponding to labels in ϑ are unchanged in the labelled ontology O', the context $O'_{l'}$ is consistent.

Otherwise, if $l' \notin L$ then, α is added to all the contexts corresponding to labels in $L \setminus \vartheta$ and the context $O'_{l'}$ is a newly created context in the labelled ontology O'. After adding α to the context $O'_{l'}$, if $O'_{l'}$ is consistent, then this context is added to O'. If $O'_{l'}$ becomes inconsistent after adding α then, Algorithm 6 calls the procedure Algorithm 3 to compute the repairs for inconsistency. Algorithm 3 returns all the repairs for inconsistency w.r.t $O'_{l'}$. Among all the repairs, only the repairs that entail the consequence α are added to O'. Thus Algorithm 6 ensures that all the contexts in O' are consistent.
2. Prove that, the context $O'_{l'}$ in the labelled ontology O' is such that $O'_{l'} \models \alpha$, Depending on ϑ, there exists two cases.
(a) If $(\vartheta = L)$ then, it implies that the consequence α is entailed from all contexts of O. Thus, it is not added to any context of O and in this case, the set of labels $L' = L$. Thus, the contexts in O' are equivalent to the contexts in O. Thus, any context $O'_{l'}$ in O' entails the consequence α.
(b) If $(\vartheta \subset L)$ then, it implies that, α holds from only few contexts of O corresponding to the labels that belong to the boundary ϑ. If the label $l' \in \vartheta$ then, α is not added to the

Algorithm 6 Remove inconsistency from contexts of the ontology

procedure RemoveIncosistencyFromContexts(O, α)
  Input: O: labelled ontology, α: the consequence to be added to every context of the output labelled ontology O'
  Output: O': labelled ontology in which every context is consistent and entails α
  Global: O':=$\emptyset$, L: set of labels in the ontology O, L':= $\emptyset$; set of labels in the ontology O', $\vartheta_\perp$:=$\emptyset$, $O_{\vartheta_\perp}$:=$\emptyset$, $\mathcal{R}_{\vartheta_\perp}$:=$\emptyset$
  $\vartheta_\perp \leftarrow$ *compute–boundary*(O, α)
  if $\vartheta_\perp$=$\emptyset$ **then**
    for every axiom, $a \in O$ **do**
      $O' \leftarrow O' \cup \{a\}$
    return(O')
  else
    for every $l \in \vartheta_\perp$ **do**
      for every $a \in O_l$ **do**
        $O_{\vartheta_\perp} \leftarrow O_{\vartheta_\perp} \cup \{a\}$
    for every $l \in (L \setminus \vartheta_\perp)$ **do**
      for every axiom, $a \in O_l$ **do**
        $O' \leftarrow O' \cup \{a\}$
      $L' \leftarrow L' \cup l$
    $\mathcal{R}_{\vartheta_\perp} \leftarrow ComputeBoundaryRepairs(O_{\vartheta_\perp}, \alpha, \vartheta_\perp)$
    for every $R_j \in \mathcal{R}_{\vartheta_\perp}$ **do**(where $1 \leq j \leq |R_{\vartheta_\perp}|$)
      for every $l \in L'$ **do**
        if $R_j \subseteq O'_l$ **then**
          $\mathcal{R}_{\vartheta_\perp} \leftarrow \mathcal{R}_{\vartheta_\perp} \setminus R_j$
        else
          $l_{new} \leftarrow n$ (where $n \in N$ and $n \notin L'$)
          $L' \leftarrow L' \cup \{l_{new}\}$
          if $R_j \models \alpha$ **then**
            for every axiom, $a \in R_j$ **do**
              if $a \in O'$ **then**
                $O' \leftarrow O' \setminus \{a\}$
              $lab(a) \leftarrow lab(a) \cup \{l_{new}\}$
              $O' \leftarrow O' \cup \{a\}$
  return(O')
end procedure

contexts $O_{l'}$. As $O_{l'}$ entails the consequence α, it is unchanged in the ontology O'. As every context in the ontology O is consistent, every context in the ontology O' whose labels are present in ϑ are consistent and entail the consequence α.

Otherwise, if $l' \notin \vartheta$ then, α is added to all the contexts corresponding to labels in $L \setminus \vartheta$. After adding α to a context, $O_{l'}$, if $O_{l'}$ is consistent, then this context is added to O'. If $O_{l'}$ becomes inconsistent after adding α then, Algorithm 6 calls the procedure in Algorithm 3 to compute repairs for inconsistency as Algorithm 3 returns all repairs for inconsistency w.r.t $O_{l'}$. Among the repairs obtained from Algorithm 3, only those repairs that entail the consequence α will be added to O'. Thus Algorithm 5 ensures that all the

contexts in O' entails the consequence α. □

Lemma 3.16. *Given a labelled ontology O in which every context is consistent and a consequence α, Algorithm 5 returns the labelled ontology O' in* exponential time*, such that every context O' is consistent and every context of O' entails a given consequence α.*

Proof : From lemma 2.11, we know that, it takes *exponential time* to compute boundary for a given consequence w.r.t to a given ontology. After computing boundary and adding the consequence to contexts from which it does not hold. If any context of the ontology becomes inconsistent, then, it takes *exponential time* to compute repairs for inconsistency as shown in lemma 1.7. Thus, the overall complexity of Algorithm 5 is *exponential time.* □

This section presented the theoretical methods developed for context-based ontology revision operation. Overall, in this chapter we presented the theoretical methods developed to do the context-based ontology evolution operations. The next chapter, presents the details about extraction of the best contexts from the labelled ontology and context-based reasoning methods over the best contexts extracted.

Chapter 4

Extraction Of The Best Contexts And Reasoning Over The Best Contexts

Chapter 3 presented the theoretical methods to do context-based ontology evolution operations on semantic web ontologies. While doing iterative ontology evolution on an ontology using context-based ontology evolution operations, instead of choosing the optimal solution after each update on the ontology, generally all the solutions obtained in every evolution step are stored as a single ontology and further evolution is done on that ontology (which is compact representation of all the solutions obtained). But, at the end of all the evolution steps, it is important to choose the optimal solution among all the solutions obtained.

As every solution is represented as a context in the labelled ontology obtained after context-based ontology evolution operations, choosing the optimal solution means, extracting a context which has some special properties. This chapter focuses on various aspects involved in extraction of a context from a labelled ontology and also on context-based reasoning over the extracted contexts. It presents the theoretical methods to extract the best contexts from a labelled ontology and methods to do context-based reasoning over the extracted contexts.

4.1 Extraction Of The Best Context

There are mainly two aspects to be considered in the process of extracting a context from a labelled ontology. They are:

1. Which context should to extracted from a labelled ontology?
2. When to extract a context from a labelled ontology?

This section discusses these two aspects of extraction in detail.

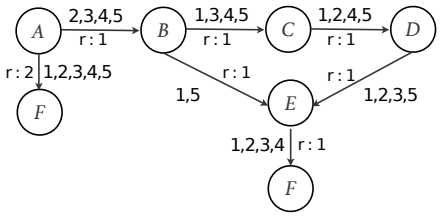

Figure 4.1: Ontology labelled with contexts and ranks

4.1.1 Notion Of The Best Context

There exists many contexts in a labelled ontology, among them, the optimal solution for the context-based ontology evolution process is, the best context (context with some special properties) in the ontology. There is no fixed notion for the best, the best context can be a context with highest number of axioms or, a context with most original axioms of the ontology or, a context with maximum weight or, a context with a intended consequence or, the context with highest number of axioms and entails a intended consequence. In our work, we consider three notions for the best: *size, rank, the best context with a intended consequence.*

Size

Here *size* refers to the number of axioms in a context. The best context w.r.t size is the context of the labelled ontology which contains highest number of axioms. This notion for the best can be considered when the optimal solution of the ontology evolution process is considered as the solution with *minimal change* [5]. As the best context obtained using this notion for the best is, the context which maximally preserves the semantics of the original ontology. We consider this notion for the best in context-based ontology contraction, revision and expansion operations. The following example shows how to extract the best context from a labelled ontology when the notion for the best is size.

Example 4.1. *Consider the labelled ontology O, shown in Figure 4.1. There are five contexts in the ontology O and the task is to extract the context with highest number of axioms from this labelled ontology.*

To extract the best context w.r.t size, sizes of all the contexts should be computed.

sizes *of the five contexts (O_1, O_2, O_3, O_4, O_5) present in the labelled ontology O are:*
$| O_1 |$ *= 5 (as the number of axioms in the context, O_1 are five);*
$| O_2 |$ *= 4;*
$| O_3 |$ *= 4;*
$| O_4 |$ *= 4;*
$| O_5 |$ *= 5;*

The biggest size among all the contexts of the ontology O is, $max_{size} = 5$. Among the five contexts in the ontology O, there are two contexts with size 5, they are, O_1 and O_5. Thus, both the contexts O_1 and O_5 are the best contexts of O w.r.t size.

In the above example, there are two best contexts in the labelled ontology (i.e., two optimal solutions) and it is the choice of the user which context to extract from the best contexts. It is also possible to extract all the best contexts and represent them as a single labelled ontology.

Rank

The best context w.r.t *rank* is the context of the labelled ontology with highest rank. That is, the context which contains highest ranked axioms of the ontology. In Chapter 3, it is shown, how the ranks are assigned to the axioms of the ontology. The original axioms of the ontology are assigned with the highest rank "1". The new axioms added to the contexts of the ontology are assigned with a lower rank than the original axioms (i.e., value of rank for a newly added axiom will be greater than 1). That is, the best context w.r.t rank is the context of the ontology which has the most original axioms of the ontology. We consider this notion for the best in context-based revision and expansion operations, as the ranks are assigned to the axioms of the ontology, only when the new axioms are added to the ontology. This notion for the best is used when the optimal solution of ontology evolution process is chosen as the solution with most original axioms of the ontology. The following example shows, how to extract the best context w.r.t "rank" from a labelled ontology.

Example 4.2. *Consider the labelled ontology O shown in Figure 4.1. The task is to extract the best context w.r.t rank from this labelled ontology.*
In the labelled ontology O, there are five contexts and ranks are assigned to all the axioms of the ontology. To extract the best context w.r.t rank, first ranks of all the contexts should be computed.

*The **ranks** of the contexts are:*
$rank(O_1)$ = *2 (as the highest rank among all the axioms of* O_1 *is 2);*
$rank(O_2)$ = *2;*
$rank(O_3)$ = *2;*
$rank(O_4)$ = *2;*
$rank(O_5)$ = *2;*

After computing ranks of all the contexts, the highest rank among all the contexts is computed, The highest rank among all the contexts is "2". As all the context have highest rank, all contexts of the ontology are the best w.r.t rank. Thus, any context can be chosen as the best context of this ontology.

As described in the case of *size*, in the case of *rank* also, the user can extract either one of the best context or all the best contexts from the ontology.

Best Context With A Intended Consequence

Extraction of the best context using this notion for the best is done in two steps.

1. Extract all the best contexts from the labelled ontology w.r.t "size" or "rank".
2. Extract a context among the best contexts extracted in step 1, which entails the intended consequence specified by the user.

We consider this notion in context-based contraction, revision and expansion operations. This notion for the best is very useful if the user wants a best context from the ontology with a intended consequence. The following example shows, how to extract the best context w.r.t the notion "Best context with a intended consequence" from a labelled ontology.

Example 4.3. *Consider the labelled ontology O shown in Figure 4.1. The task is to extract the best context w.r.t the notion, "Best context with a intended consequence" from this labelled ontology.*
Step 1: *Extract all the best contexts from the labelled ontology, let the notion for the best be size. The best contexts in the ontology O w.r.t size are* O_1 *and* O_5 *(shown in Example 4.1). Let the consequence* α*:* $A \sqsubseteq B$ *be the intended consequence from O.*
Step 2: *From the two best contexts computed in step 1, the context which entails the consequence* $A \sqsubseteq B$ *should be extracted.*

The best contexts contain the following axioms:
$O_1 = \{B \sqsubseteq C,\ C \sqsubseteq D,\ D \sqsubseteq E,\ B \sqsubseteq E,\ E \sqsubseteq F\}$ *and*
$O_5 = \{A \sqsubseteq B,\ B \sqsubseteq C,\ C \sqsubseteq D,\ D \sqsubseteq E,\ B \sqsubseteq E\}$.
Among the contexts O_1 *and* O_5, $O_1 \not\models A \sqsubseteq B$ *and* $O_5 \models A \sqsubseteq B$. *Thus, the best context is* O_5.

If there exists more than one best context in the ontology according to this notion then, any one of them can be chosen as best or all the best contexts can be extracted and represented as a single labelled ontology. If there exists no contexts in the ontology, which satisfy this notion for the best then, the best context computed at the end of first step is extracted as the best context of the ontology. Thus, we consider these three notions for the best and present the theoretical methods to extract the best contexts from a labelled ontology w.r.t these three notions for the best.

Here we present two algorithms to extract the best context from a labelled ontology. The procedure described in *Algorithm* 7 is used to extract the best context(s) from the ontology, when the notion of the *best* is "size" or "rank". The algorithm takes a labelled ontology O, the notion of the best ("size" or "rank"), Number of solutions (whether the user wants to extract all the optimal solutions or only one optimal solution) as input. If the notion of the best is "size" then, this algorithm computes sizes of all the contexts of the ontology in steps 6-10 and stores the sizes in the set *sizes*. After computing sizes of all the contexts, the algorithm computes the maximum size among the sizes of all the contexts in step 11 and stores it in the variable max_{size}. Then it extracts all the contexts of the ontology with max_{size} from the ontology and stores them in the set $O_{AllBest}$ in step 14.

If the notion for best is "rank" then, the algorithm first computes the ranks of all the axioms that belong to a context and stores these ranks in the set $rank(O_l)$ in step 19. Among the ranks of all the axioms in a context, the rank of the context is computed as the minimum value of all the ranks. The rank of a context is stored in the variable $rank_l$ (where l is the label of the context O_l), this computation is done in step 20 and ranks of every context are stored in the set *ranks* in step 21. After the ranks of all the contexts are computed, the algorithm then computes the highest rank among all the contexts, this is done in the step 22, this value is stored in the variable max_{rank}. Then, in the step 25 the algorithm extracts all the contexts $O_{AllBest}$ from the ontology O whose rank is max_{rank}.

After extracting all the best contexts from the labelled ontology w.r.t "size" or "rank", if the user wants to extract all the best contexts from the ontology, then a single labelled ontology O_{Best} is created using all the axioms that belongs every best context present in the set $O_{AllBest}$, this computation is done in step 29. If the user wants only one optimal solution then a random context from the set of contexts in $O_{AllBest}$ is taken as the best context O_{Best} in step 32. The algorithm returns the best context O_{Best} as output in step 33. Thus the algorithm extracts all the best contexts or a single best context from the given labelled ontology.

Extraction of the best context with a intended consequence α from the labelled ontology requires two steps. The first step is to extract all the best contexts of the ontology w.r.t "size" or "rank". The second step is to extract a context from the set of best contexts, which entails a intended consequence α. *Algorithm 8* describes the step by step procedure to extract the best context that entails a consequence α from a labelled ontology.

Algorithm 7 Extract best contexts from the labelled Ontology

1: **procedure** ExtractBestContext(O, *Best*, *numberOfOptimalSolutions*)
2: **Input:** O: labelled ontology, *Best*: notion for the best, *numberOfSolutions*: number of optimal solutions to be extracted from O
3: **Output:** O_{Best}: labelled ontology containing extracted best contexts
4: **Global:** L: set of labels in O, $ranks := \emptyset$, $sizes := \emptyset$, $O_{AllBest} = \emptyset$; set which contains all the best contexts, $O_{Best} = \emptyset$; labelled ontology containing extracted optimal solutions
5: **if** *Best* = "size" **then**
6: **for** every $l \in L$ **do**
7: $size_l \leftarrow 0$;
8: **for** every $a \in O_l$ **do**
9: $size_l \leftarrow size_l + 1$;
10: $sizes \leftarrow sizes \cup size_l$
11: $max_{size} \leftarrow maximum(sizes)$ ▶ max_{size} the maximum value in the set, $sizes$.
12: **for** every $l \in L$ **do**
13: **if** $size_l = max_{size}$ **then**
14: $O_{AllBest} \leftarrow O_{AllBest} \cup O_l$
15: **else if** *Best* = "rank" **then**
16: **for** every $l \in L$ **do**
17: $rank_l \leftarrow 0$;
18: **for** every $a \in O_l$ **do**
19: $rank(O_l) \leftarrow rank(O_l) \cup rank(a)$ ▶ $rank(a)$ is the rank of the axiom, a
20: $rank_l \leftarrow$ minimum($rank(O_l)$) ▶ minimum value in $rank(O_l)$ is the highest rank of O_l
21: $ranks \leftarrow ranks \cup rank_l$
22: $max_{rank} \leftarrow minimum(ranks)$ ▶ minimum value in *ranks* is the highest rank in the ontology
23: **for** every $l \in L$ **do**
24: **if** $rank_l = max_{rank}$ **then**
25: $O_{AllBest} \leftarrow O_{AllBest} \cup O_l$
26: **if** *numberOfSolutions* = "all" **then**
27: **for** every $O_i \in O_{AllBest}$ **do** (where $1 \leq i \leq |O_{AllBest}|$)
28: **for** every axiom, $a \in O_i$ **do**
29: $O_{Best} \leftarrow O_{Best} \cup a$
30: **else if** *numberOfSolutions* = "one" **then**
31: let $i = 1$;
32: $O_{Best} \leftarrow O_{Best} \cup O_i$ (where $O_i \in O_{AllBest}$)
33: return(O_{Best})
34: **end procedure**

The algorithm takes a labelled ontology O, notion for the best ("size" or "rank") and the intended consequence α as input. It returns the best context O_{Best} as output. In step 6, the algorithm calls the procedure in Algorithm 7 with the ontology O, *Best*, *numberOfSolutions* as input. As we want to check the entailment of α w.r.t all the best contexts of the ontology, the value of the variable *numberOfSolutions* is set to "all". The algorithm checks the entailment of α from every best context in the set $O_{AllBest}$ in step 8. If any of the contexts in $O_{AllBest}$ entails α then, that context is stored in O_{Best} and the

Algorithm 8 Extract the best context entailing α from the labelled Ontology

1: **procedure** ExtractBestEntailedContext(O, *Best*, α)
2: **Input:** O: labelled ontology, *Best*: notion for the best, α: consequence to be entailed from the best contexts of O
3: **Output:** O_{Best}: labelled ontology containing extracted best context
4: **Global:** L: set of labels in O,$O_{AllBest} = \emptyset$; set which contains all the best contexts, $O_{Best} = \emptyset$; labelled ontology containing extracted best context, *numberOfSolutions*=0.
5: *numberOfSolutions* $\leftarrow$ *all*;
6: $O_{AllBest} \leftarrow$ ExtractBestContext(O,*Best*, *numberOfSolutions*)
7: **for** every context $O_i \in O_{AllBest}$ **do** (where $1 \leq i \leq |O_{AllBest}|$)
8: **if** $O_i \models \alpha$ **then**
9: $O_{Best} \leftarrow O_{Best} \cup O_i$
10: return(O_{Best})
11: **if** $O_{Best} = \emptyset$ **then**
12: let $i = 1$;
13: $O_{Best} \leftarrow O_{Best} \cup O_i$ (where $O_i \in O_{AllBest}$)
14: return(O_{Best})
15: **end procedure**

algorithm returns O_{Best} as the best context of the ontology.

Otherwise, if none of the contexts in $O_{AllBest}$ entail the consequence α, then the algorithm takes a random context in $O_{AllBest}$ as the best context, this computation is done in step 12. The algorithm returns the best context extracted from the ontology in step 13. Thus, this algorithm can be used to extract the best context from a labelled ontology which has a intended consequence α.

The previous paragraphs presented the procedures to extract the best context(s) from a labelled ontology using the three notions of the best that are described in this work. The Theorem 4.4 proves the correctness of the procedure to extract best context from a labelled ontology with "size" or "rank" as the notion of the best using Algorithm 7.

Theorem 4.4. *Given a labelled ontology O, Algorithm 7 returns a context O_{Best} which is the best context of O w.r.t "size" or "rank".*

Proof : Depending on the notion for the best, we should prove the following two cases.
1. If *Best* = *size*
In the labelled ontology O, let L be the set of labels. For some arbitrary label $l \in L$, let O_l be a context with the highest number of axioms. It means that, size of the context O_l is the max_{size} in the ontology. As the context O_{Best} which Algorithm 7 returns is the context with the size max_{size}, it should be one of the best contexts of O w.r.t "size". If O_{Best} is not one of the best contexts of O, then it contradicts our assumption that, the context O_l has the highest number of axioms and the size of O_l, max_{size} is not the biggest size of a context in O. Thus, we conclude that O_{Best} is a best context in O w.r.t "size".
2. If *Best* = *rank*
The proof of case 2 is analogous to proof of case 1. Let L be the set of labels in the labelled ontology O. For some arbitrary label $l \in L$, let O_l be a context with the highest rank in

the ontology. Then max_{rank} is the rank of the context O_l. As the context O_{Best} which Algorithm 7 returns is the context with has the rank max_{rank}, it should be one of the best contexts of O w.r.t "rank". If O_{Best} is not one of the best contexts of O then, it contradicts our assumption that, the context O_l is the highest ranked context in O and the rank max_{rank} of O_l is not the highest rank of a context in O. Thus, we conclude that O_{Best} is a best context in O w.r.t "rank". □

The following Lemma 4.5 proves the complexity of the procedure described in Algorithm 7 to extract a best context from a labelled ontology.

Lemma 4.5. *Given a labelled ontology O, Algorithm 7 returns the best context from O in linear time.*

Proof : Algorithm 7 takes *linear time* to extract best context from a labelled ontology because of the following: preprocessing steps are done on an ontology while doing context-based ontology evolution operations on it using Algorithm 2 or 6. The preprocessing steps are, the algorithm maintains a hash table of every context of the ontology and the axioms belonging to that context. Thus it takes constant time to compute the size of each context in Algorithm 7. And it takes constant time to compute the max_{size} or max_{rank} among all the contexts of the ontology. Once the max_{size} or max_{rank} is computed, it takes *linear time* to enumerate all the axioms of the labelled ontology and extract a context having max_{size} or max_{rank} from the ontology. Thus, due to the prepossessing steps performed on the given ontology, it takes *linear time* to extract a best context from a labelled ontology using Algorithm 7. □

This section discussed about the notion for the best, presented in detail about the various notions for best considered in this work and presented theoretical methods to extract a best context from an ontology using the three notions for the best. It also presented the theorem to prove correctness of the algorithm and also showed the complexity of Algorithm 7. The next section handles another aspect to be considered in extraction of a best context, that is, When to extract a context from a labelled ontology?

4.1.2 When To Extract A Context From A Labelled Ontology?

With respect to this aspect, we considered two ways to extract the best context from a labelled ontology. They are:
1. Extraction of the best context during iterative context-based ontology evolution.
2. Extraction of the best context after iterative context-based ontology evolution.

Extraction Of The Best Context During Iterative Context-based Ontology Evolution

While doing iterative ontology update on an ontology, using regular ontology evolution approaches, generally the optimal solution is chosen at the end of every update and the next update are done on this solution. In the same manner, when iterative ontology update

is done on an ontology using context-based ontology evolution approaches also, it is possible to choose the optimal solution after every update and do the next update on this solution.

For example, consider an ontology O and set of consequences α_1 and α_2, in the first step, context-based ontology evolution operation is done on O w.r.t α_1. Then, a context which is the optimal solution from the first ontology evolution step, is extracted at the end of first step. Let O_{Best} be the context obtained after first step. The next step of context-based ontology evolution is done on O_{Best} w.r.t the consequence α_2. In this manner, best context is extracted at the end of every step of ontology evolution and the best context extracted is used to do the next step of context-based ontology evolution operation.

This method may lead to more information loss, as it considers only one optimal solution obtained at the end of every step of evolution. But doing context-based ontology updates using this method is fast as computations are done only on one context, instead of the whole ontology.

Extraction Of The Best Context After Iterative Context-based Ontology Evolution

The main aim of doing iterative ontology update using context-based ontology evolution methods is to store all the solutions obtained in each step of ontology evolution process compactly and do further updates on all the solutions to minimize information loss. Extracting a context from the ontology using this approach, *Extraction of best context after iterative context-based ontology evolution* enables to do iterative ontology update on all the solutions obtained at every stage of ontology evolution and a context can be extracted from the ontology at the end of all steps of ontology evolution process.

For example, consider an ontology O and a set of consequences α_1 and α_2, context-based ontology evolution operations are done on the two consequences w.r.t the whole ontology O. That is, let O' be the result of context-based ontology operation on O w.r.t α_1. In the next step, context-based ontology evolution operation is done on O' w.r.t α_2. In this manner, all the solutions obtained in every step are stored and represented as a single labelled ontology and further updates are done on it. At the end of all steps of evolution, a context can be extracted from the labelled ontology obtained after all the steps of context-based ontology evolution.

This method minimizes the information loss from the ontology. But, as it keeps the whole ontology in every step, it needs more memory to store all the solutions and more time to do computations on the whole ontology. To understand the trade-offs between these two methods, we performed experiments on real-world ontologies comparing the two methods. The results and analysis of the experiments is presented in the next chapter.

In this work, we considered these two methods discussed above. But, there is no fixed rule about when a context should be extracted from an ontology. Best contexts can be extracted from a labelled ontology at any stage during iterative ontology update.

This section elaborated on the task of extraction of the best context from a labelled ontology. It presented the notions of the best and described an algorithm how to extract the best context. It also presented various aspects to be considered during extraction. The point to be noted here is that, we can extract one best context among all the best contexts present in an ontology or we can extract all the best contexts present in the ontology as optimal solutions. The next section presents the theoretical methods to do context-based reasoning over the extracted best contexts.

4.2 Reasoning Over The Best Contexts

Context-based reasoning offers the reasoning tasks such as, checking entailment of a subsumption, checking consistency of the context in the ontology and computing boundary of a consequence w.r.t extracted contexts. To do context-based reasoning effectively over the extracted contexts, some preprocessing steps should be done on the extracted contexts. In extraction of best contexts from an ontology, a context or a set of best contexts are extracted from a labelled ontology. But the ontology contains many contexts other than the extracted contexts. Due to this, the labels of the axioms present in the extracted contexts might contain labels of contexts in the ontology other than the labels of extracted contexts. But it is not efficient to do context-based reasoning over such axioms which have un-intended labels. To overcome this problem, after extracting best contexts from an ontology, preprocessing steps are done to get rid of un-intended labels from the axioms of extracted contexts and to do efficient context-based reasoning over extracted contexts.

The steps are, to modify the labels of all the axioms of the extracted contexts such that, the axioms contains only the labels of the contexts that are extracted. All other labels are removed from the labels of the axioms. To retain only the labels of the best contexts extracted, it is important to know the labels of the best contexts. The following algorithm is used to do the preprocessing steps and context-based reasoning over the extracted contexts.

The procedure to do the preprocessing steps and context-based reasoning over the extracted contexts is described step-by step in Algorithm 9. The algorithm takes three input parameters O: labelled ontology which contains all the extracted contexts, *labels*: labels of the best contexts extracted, α: consequence to do context-based reasoning. The algorithm uses the global variables O': ontology created using the relabelled axioms of the ontology O, ϑ: stores the boundary computed for α w.r.t O'. The algorithm first processes the labels of the axioms of the input ontology. It creates new labels to the axioms $lab_{new}(a)$ by retaining only the labels of best contexts of the axioms, i.e., labels present the variable *labels*, this is done in steps 7,8. It removes all other labels from the axioms and creates the ontology O' with the relabelled axioms in step 9.

After the ontology is relabelled then, context-based reasoning can be done over it. It is done in the step 10, that is, we can compute boundary of the input consequence α w.r.t the relabelled ontology O'. Finally in step 11, the algorithm returns the boundary ϑ as the result of context-based reasoning over the extracted ontology.

Algorithm 9 Context-based reasoning over the best contexts

procedure ReasoningOverTheBestContexts(O, *labels*, α)
  Input: O: labelled ontology, *labels*: context labels w.r.t which the ontology O is extracted, α: consequence to do context-based reasoning
  Output: ϑ: boundary of the consequence α
  Global: O':= $\emptyset$; stores relabelled axioms of the input ontology, ϑ:= boundary of α w.r.t O'.
  for every axiom, $a \in O$ **do**
    $lab_{new}(a) \leftarrow \emptyset$
    $lab_{new}(a) \leftarrow lab(a) \cap labels$
    $lab(a) \leftarrow lab_{new}(a)$ ▶ remove old label of the axiom a and assign new label
    $O' \leftarrow O' \cup \{a\}$
  $\vartheta \leftarrow compute\text{–}boundary(O', \alpha)$
  return(ϑ)
end procedure

The complexity of Algorithm 9 is same as the complexity of computing boundary. The complexity of computing boundary in context-based reasoning is *exponential time*, which is proved in the lemma 2.11.

This chapter presented the theoretical methods to extract best contexts from a labelled ontology and to do context-based reasoning over the extracted contexts. It also discussed various aspects to be considered while extracting the best contexts from an ontology like the notion of the best and when to extract a context from an ontology. The next chapter presents the implementation details of the theoretical methods developed in Chapters 3 and 4. It also presents the details about the empirical evaluation done to evaluate our implementation on the real-world ontologies.

Chapter 5

Implementation and Empirical Evaluation

All the theoretical methods to do context-based ontology evolution operations, to extract the optimal solutions and to do context-based reasoning over the extracted optimal solutions are described in Chapters 3 and 4. This Chapter describes the first prototypical implementation of all the theoretical methods developed in Chapters 3 and 4. It also presents the protége plug-in developed to do context-based ontology evolution operations. Section 5.2 presents the details of the experiments on the real-world ontologies FULL-GALEN[1] and SNOMED CT[2] to evaluate the implementation of our approaches. The last section of this chapter presents the results of our experiments, interpretation of the results and our conclusions based on the results.

5.1 Implementation

All the approaches are implemented in Java 7. The logic reasoners Hermit[3] and ELK[4] are used in this work. Hermit is an OWL 2 DL reasoner based on hyper-tableau calculus. Given an ontology, it can perform consistency checking, classification, entailment checking and also gives justifications for an entailment. ELK is a reasoner for the lightweight ontology language OWL 2 EL. It is very fast and supports incremental reasoning. The OWL API[5] is used, as it is convenient for parsing OWL ontologies and to interact with the reasoners. The experiments are conducted on a PC with 2GB RAM and Intel Core Duo CPU 3.16GHz. A Java swing application is developed as a plug-in to the ontology editor tool Protége 4.3.[6]

[1] http://www.opengalen.org
[2] http://www.ihtsdo.org/snomed-ct/
[3] http://www.hermit-reasoner.com
[4] https://code.google.com/p/elk-reasoner/
[5] http://owlapi.sourceforge.net/
[6] http://protege.stanford.edu

5.1.1 Protége Plug-in

The plug-in to Protége 4.3 is named as *Context-based Ontology Evolution Plug-in*. The main aim of this plug-in is to provide the user, a graphical interface to do context-based ontology evolution operations on the ontologies loaded into the tool. Figure 5.1 shows the main view of the plug-in. As soon as the plug-in is loaded, it displays the information about the contexts present in the ontology. The plug-in can handle context-based ontology contraction and context-based ontology revision operations.

To do context-based evolution operations, the user should enter the concepts names of super class and sub class of the subsumption relation on which the user wants to do the operations. With this information the user can do the following operations: check, if the ontology entails the consequence, check boundary of the consequence w.r.t the ontology loaded, context-based ontology contraction operation, context-based ontology revision operation, extracting the best context that entails the given consequence from the given labelled ontology and the user can also extract some random context from the ontology.

The button "Check Entailment" checks, if the given consequence is entailed from the ontology. The button, "Boundary" computes the boundary of the given consequence w.r.t the labelled ontology that is loaded into Protége. The button, "Context-based Ontology Contraction" is used to do context-based ontology contraction operation on the given consequence. The button "Context-based Ontology Revision" is used to do context-based ontology revision operation on the given consequence. The button "Extract Best Context" is used to extract one of the best context w.r.t "size" or "rank" from the labelled ontology. And the button, "Extract A Context" extracts a context from the labelled ontology, given the label of a random context in the ontology. Thus, the user can do context-based reasoning and context-based ontology evolution operations using this plug-in.

5.2 Empirical Evaluation

This section describes in detail about the experiments we conducted, the ontologies and test setting used for the experiments. Among the theoretical methods developed, our experiments evaluated the algorithms *context-based ontology contraction* which is the implementation of Algorithm 2 and *extract the best context* which is the implementation of Algorithm 7.

5.2.1 Ontologies

The real-world ontologies FULL-GALEN and SNOMED CT with different expressivities are chosen for our experiments. The Systematized Nomenclature of Medicine, Clinical Terms (SNOMED CT) is a comprehensive medical and clinical ontology whose expressivity is $\mathcal{EL}^{++}$ and FULL-GALEN ontology with expressivity is $\mathcal{ALCHIF}$. The metrics of both the ontologies are shown in Figure 5.2. For our experiments we chose the consequences with

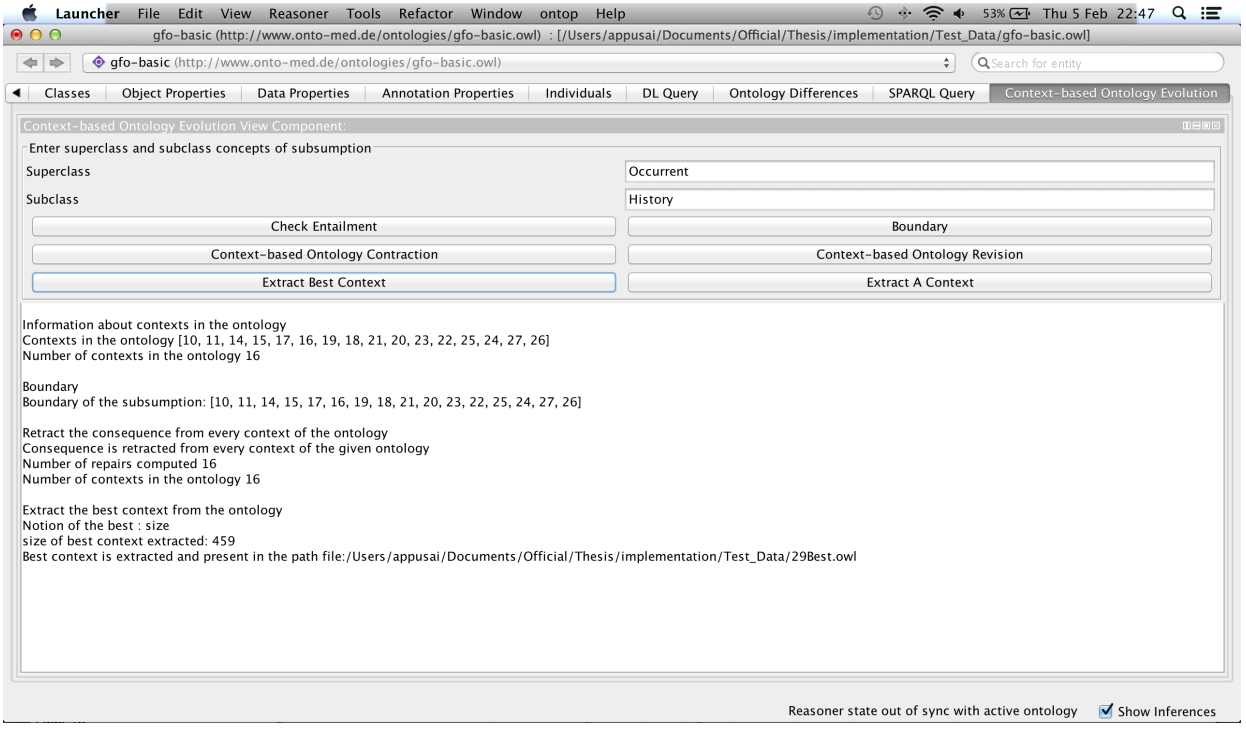

Figure 5.1: Context-based Ontology Evolution plug-in in Protége 4.3

Ontology Name	Number of axioms	Number of concept names	Number of object properties
FULL-GALEN	63329	23141	950
SNOMED CT	883,542	294,469	62

Figure 5.2: Metrics of FULL-GALEN and SNOMED CT ontologies

the following properties: the consequences with more MinAs and consequences with few MinAs. As the number of MinAs increase for a consequence, the number of repairs for that consequence also increases. In our tool, as the number of contexts and the boundary size for the consequence increase, the number of repairs computed for the consequence w.r.t every context in the boundary also increases. Thus, the consequences with these properties pose different challenges to our tool. All the consequences for SNOMED CT testing are chosen from the test data used in Debugging Large $\mathcal{EL}+$ Ontologies via SAT and SMT Techniques [16]. All the consequences for FULL-GALEN testing are chosen from the test data used in testing Just Reasoner [17].

5.2.2 Test Setting

Every test case designed for our experiments consists of an ontology in which all the axioms are labelled with a single context label "1" and five consequences to be tested on the ontology. The representation of a test case is shown in Figure 5.3.

The main focus of our experiments is to compare the performance of two scenarios which are described in Chapter 4. These two scenarios are chosen as they show how information loss can be minimised in iterative ontology update by storing all the solutions obtained in every step of ontology update. Scenario 1 is called *"extraction of the best context after iterative ontology update"*. This scenario is represented in Figure 5.4. There are five steps in this scenario. In every step, the operation (either context-based ontology contraction or context-based ontology revision) is performed on the whole ontology that is given as input. After doing context-based ontology evolution operation on the first four consequences of the test case, a labelled ontology is obtained. Using this labelled ontology and the fifth consequence of the test case, the best context (w.r.t size or rank) which entails the fifth consequence of the test case is extracted from the labelled ontology. The context extracted is the result of scenario 1.

The second scenario (scenario 2) is called "extraction of the best context during iterative ontology update". This scenario is represented in Figure 5.5. The figure shows that, two operations are performed on every consequence of the test case. They are, context-based ontology evolution operation (either contraction or revision) and then extraction of the best context (w.r.t size or rank) from the resulting labelled ontology. The best context extracted at the end of every step is used to do the two operations mentioned above on the next consequence of the test case. Thus, the operations context-based ontology evolution operation and extraction of the best context are done on the first four consequences of the test case. At the end of four steps, we test if the best context extracted at the end of four steps entails the fifth consequence of the test case and then extract the best context.

Our hypothesis is that, the running time of scenario 1 is more than the running time of scenario 2. As scenario 1 uses the whole ontology in every step and it requires more memory to store the whole ontology and more time to do computations on all axioms of the ontology. But it is still possible to keep whole ontology and do operations over the whole ontology in reasonable time using context-based ontology evolution approaches. Scenario 2 is relatively less expensive than scenario 1 as it keeps only the best context of the ontology in memory at every step and performs the computations on it. But according to our hypothesis, the scope for information loss is very low in scenario 1 than in scenario 2 as the updates are performed on the whole ontology (keeping all the solutions obtained) in every step in scenario 1. Thus, the best context extracted at the end of all the steps in scenario 1 will have equal or more information (axioms) than the best context extracted at the end of all steps of scenario 2. That is, context-based ontology evolution approach provides us an opportunity where the whole ontology can be represented and stored compactly and computations can be done on the whole ontology efficiently.

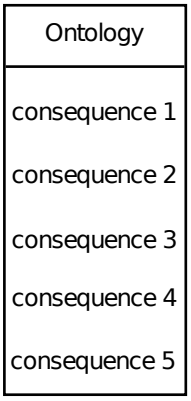

Figure 5.3: Representation of a test case

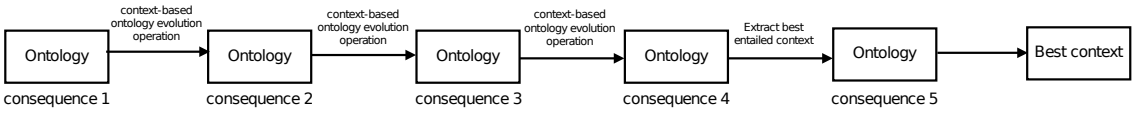

Figure 5.4 Representation of scenario 1: extraction of the best context after iterative ontology update scenario

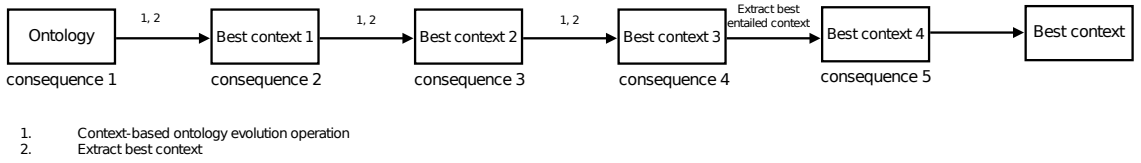

Figure 5.5 Representation of scenario 2: extraction of the best context during iterative ontology update

5.2.3 Experimental Results And Interpretation

This section presents the results of our experiments on the ontologies FULL-GALEN and SNOMED CT. On FULL-GALEN ontology we tested 20 test cases, each test case consisting of 5 consequences. Every test case is run three times and average of the three runs is taken as the result. For SNOMED CT testing, 10 test cases are considered, each test case consisting of 5 consequences. In case of SNOMED CT testing also, every test case is run three times and the average of the three runs is taken as the result. Every test case is tested in both the scenarios 1 and 2.

5.2.4 Analysis of Size of The Best Context Extracted

Figure 5.6 shows the results of comparison between the size of the best context extracted in the scenarios 1 and 2. Figure 5.6 (A) shows the results of FULL-GALEN and Figure 5.6 (B) shows the results of SNOMED CT. From the results obtained, it is observed that, in case of FULL-GALEN, in scenario 1, the maximum size of a best context extracted

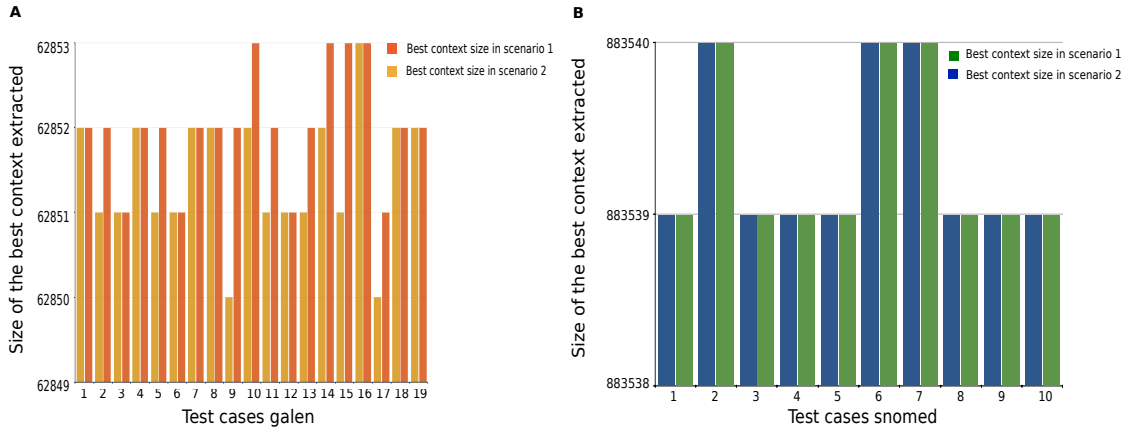

Figure 5.6 The grouped bar graphs show the comparison between size of the best context extracted in each test case in scenarios 1 and 2 for the ontologies (A) FULL-GALEN and (B) SNOMED CT

is 62853 axioms and the minimum size of a best context is 62851 axioms. In scenario 2, the maximum size of a best context extracted is 62853 and minimum size is 62850. From Figure 5.6 (A) it is clearly depicted that, the size of the best context extracted is same or more in case of scenario 1 than in scenario 2. That is, the information loss in scenario 1 is less than the information loss in scenario 2. It means that, our goal to achieve *"Iterative Ontology Update with Minimum Change"* using context-based ontology evolution approach is successful. In case of SNOMED CT results, the results are the following. In scenario 1 and scenario 2, the maximum size of a best context extracted is 883540 and minimum size is 883539. In both the scenarios the size of the best context extracted is the same.

5.2.5 Analysis of Time Taken To Run The Test Cases

The graphs in Figure 5.7 shows the comparison between total time taken to run each test case in scenario 1 and scenario 2. Figure 5.7 (A) shows the results of FULL-GALEN and Figure 5.7 (B) shows the results of SNOMED CT. For FULL-GALEN, the average time taken to run the whole test case in scenario 1 is 17.49 minutes and average time taken to run the whole test case in scenario 2 is 6.66 minutes. In case of SNOMED CT, the average time taken to the whole test case in scenario 1 is 14.44 minutes, in case of scenario 2, it is 6.75 minutes. In case of both the ontologies, average time to run the test case in scenario 1 is more, when compared to the average time to run the test case in scenario 2. This is because, in scenario 1 we keep the whole ontology at every step and do computations on the whole ontology which requires more space and time for computation. The highest number of contexts in a labelled ontology in scenario 1 for the ontology FULL-GALEN is 400 contexts. For SNOMED CT highest number of contexts is 39. In case of scenario 2, we extract the best context at the end of every step and do further ontology updates only on the extracted context, whose size is less than the whole ontology and also the number of contexts in the extracted context is always 1, which is less than the number of

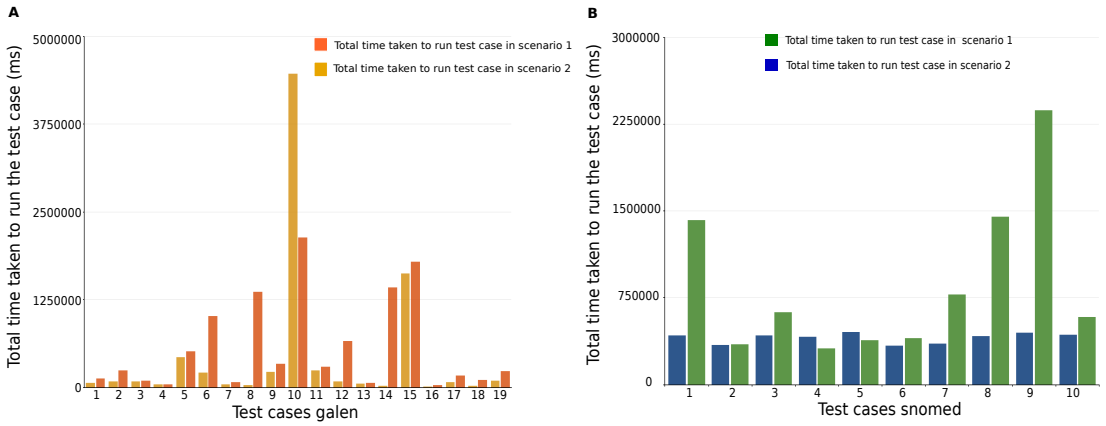

Figure 5.7 The grouped bar graphs show the comparison between the total time taken to run each test case in scenarios 1 and 2 for the ontologies (A) FULL-GALEN and (B) SNOMED CT

contexts in the whole ontology and this makes the computation of repairs simple in case of scenario 2. That is, as the number of contexts increase in the ontology, it is very likely that the boundary size for a consequence w.r.t that labelled ontology also increases. As the boundary size increases, repairs needs to be computed for the consequence w.r.t every context present in the boundary.

In case of FULL-GALEN ontology, the highest number of repairs computed for a consequence w.r.t every context present in the boundary, in case of scenario 1 is 396, for the same consequence, in case of scenario 2, the number of repairs computed is 33. As, in case of scenario 2 there is always one context in the ontology (as only one best context is extracted from the ontology after every step of evolution). Thus, increase in the number of contexts in case of scenario 1 in turn increases the number of repairs computed for a consequence w.r.t every context in the boundary. This increases the overall time taken to run a test case in case of scenario 1.

The analysis of the graphs in Figures 5.6 (B) and 5.7 (B) show that, in case of scenario 1 and scenario 2, the size of the best context extracted is the same in each test case, but the time taken to run a test case is more in case of scenario 1 than the time taken to run a test case in scenario 2. It shows that, our algorithms should be improved to improve the performance of the tool. To understand precisely the reason for more computation time in case of scenario 1, we further analysed the results of our experiments.

5.2.6 Analysis on the Performance of Context-based Ontology Contraction Algorithm

Figure 5.8 shows the dependency of the total time taken to compute repairs for a consequence on the boundary size of the consequence (i.e., number of contexts present in the boundary of the consequence) and the number of repairs computed for the consequence

Figure 5.8 The bubble graphs show the dependency of the time taken to compute repairs for a consequence, on the boundary size and the number of repairs computed, in scenario 1 for the ontologies (A) FULL-GALEN and (B) SNOMED CT

w.r.t every context present in the boundary, in the context-based ontology contraction approach. The graph in Figure 5.8 (A) represents results of FULL-GALEN and Figure 5.8 (B) represents the results of SNOMED CT. Both the graphs clearly depict that total time taken to compute the repairs for a consequence w.r.t to every context present in the boundary increases with the size of the boundary and also increases with the number of repairs computed for the consequence.

This is the main reason for more computation time in case of scenario 1 than the computation time in case of scenario 2. Unlike in scenario 1, in scenario 2 there exists only one context at every stage of ontology update from which repairs should be computed. Thus, time taken to compute repairs in scenario 2 is less than in scenario 1. In order to improve the performance of the tool, we further analysed the results. The following paragraphs present our results and analysis.

In context-based ontology contraction approach, there are two main processes. They are, computing the boundary for the consequence and computing repairs for the consequence w.r.t every context in the boundary. To improve the performance, it is essential to pinpoint which of the two processes consumes more time. Figure 5.8 shows the distribution of the time taken to compute the boundary and the time taken to compute the repairs in the total time taken to compute the test case in case of scenario 1. Figure 5.9 (A) represents the results for the ontology FULL-GALEN and Figure 5.9 (B) represents the results for the ontology SNOMED CT. In case of results for FULL-GALEN, the time taken to compute boundary is considerably less than the time taken to compute the repairs. In case of results for the ontology SNOMED CT, the time taken to compute boundary is very less than the time taken to compute the repairs. This results interpret that, the process of computing repairs for the consequence w.r.t every context in the boundary is more time consuming and this should be analysed further.

To analyse the time taken by the process, computing repairs for the consequence w.r.t

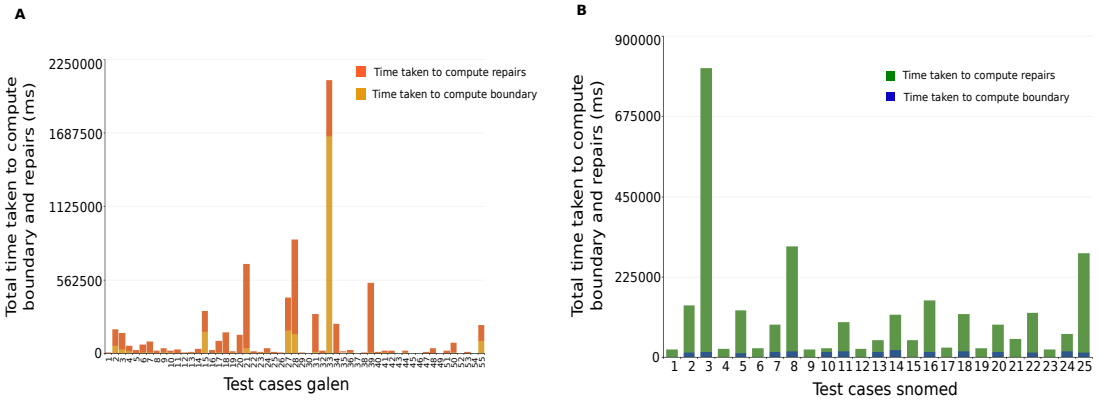

Figure 5.9 The stacked bar graphs show the division of total time taken to run the case between time taken to compute the boundary and the time taken to compute the repairs for a consequence, in scenario 1 for the ontologies (A) FULL-GALEN and (B) SNOMED CT

every context in the boundary, it is required to understand the sub-processes involved in this process. There are two sub-process involved, first sub-process is to call the reasoner to get justifications for the consequence w.r.t all the axioms present in every context that belongs to the boundary and to compute repairs from the explanations obtained. The second sub-process is to process the repairs obtained to compute the repairs w.r.t every context present in the boundary and to re-label the axioms that belong to each repair accordingly (as shown in Algorithm 3).

Figure 5.10 shows the results of the distribution of time taken by the reasoner in the total time taken to compute the repairs. Figure 5.10 (A) represents the results of FULL-GALEN, this graph shows that, the reasoner takes significant amount of time in the total time taken to compute the repairs. This performance can be improved by pluging-in more efficient reasoner which can compute the justifications for a consequence in less time (as our approach is a black-box approach we can plug-in any reasoner which provides incremental reasoning and explanation services). Figure 5.10 (B) represents the results of SNOMED CT, this results clearly show that in case of SNOMED CT, the time taken by the reasoner is negligible in total time to compute the repairs. This is because, the module sizes in case of SNOMED CT are very less than the module sizes in case of FULL-GALEN. In the test cases we used, the module sizes in SNOMED CT ontology are between 20 - 480 axioms, the module sizes in FULL-GALEN are between 22306 - 34622 axioms. In our approach, after computing each repair, the axioms of the ontology are relabelled. As the number of axioms in SNOMED CT is more than the number of axioms in FULL-GALEN, it takes more time to relabel the axioms.

From this results, we conclude that, our approach to compute repairs and relabel the axioms should be improved, to improve the overall performance of our tool. One the main reasons for more time taken to compute repairs and re-label axioms is that, all the algorithms are implemented in Java, which is very memory expensive and has bad garbage

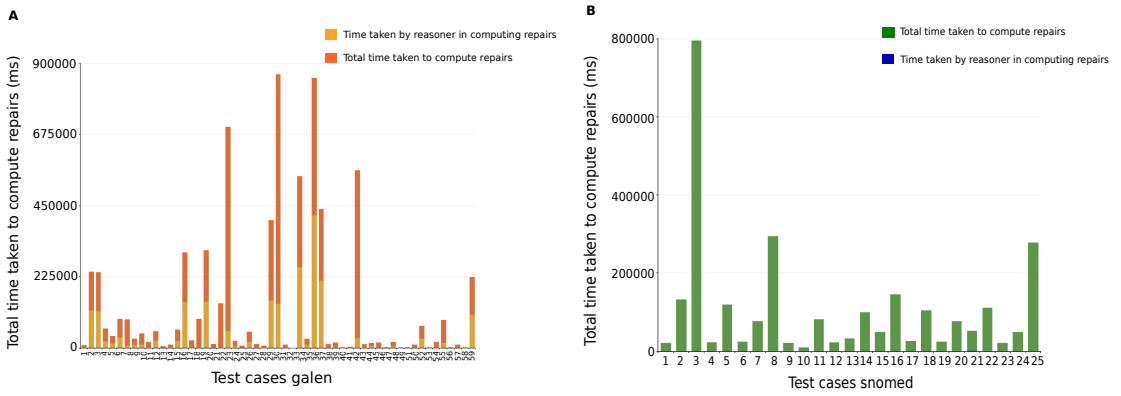

Figure 5.10 The stacked bar graphs show the proportion of time taken by the reasoner in the time taken to compute repairs for a consequence, in scenario 1 for the ontologies (A) FULL-GALEN and (B) SNOMED CT

collection. It requires high memory resources to store all the axioms and their labels. To improve the performance of the algorithm, we should find a new method to re-label the axioms of the ontology, which uses less memory resources. Thus, we evaluated the algorithms developed in this thesis and showed that our implementation works well on real-world ontologies.

This chapter presented the details of the first prototypical implementation of the theoretical methods developed in this thesis: context-based ontology contraction algorithm, context-based ontology revision algorithm, algorithm to extract the best contexts from a labelled ontology. It also describes in detail about the protége plug-in developed for handling context-based ontology evolution tasks and about the experiments conducted to evaluate the tool. This chapter also gives the details of the results obtained from the experiments conducted on the real-world ontologies FULL-GALEN and SNOMED CT. Our experiments prove that our implementation works well on real-world ontologies. It also gives in-depth interpretation of the results and analyses the ways to improve the performance of the tool.

Chapter 6

Summary and Future Research

In this work we provided a combination of ontology evolution operations with context-based reasoning methods to achieve updates on the ontology iteratively with minimum information loss. The main contributions can be listed as follows. We provided a context-based reasoning extension to the ontology evolution operations, that is different from existing approaches. With this approach, we can store all the possible solutions obtained in an ontology evolution task compactly, as a single labelled ontology. Updates and reasoning can be effectively done over the labelled ontology. We showed, how to create a labelled ontology using the solutions obtained in an ontology evolution task.

We developed theoretical methods to do ontology contraction, ontology expansion and ontology revision using context-based reasoning and methods for extracting the optimal solutions from the labelled ontology obtained as result of context-based ontology evolution operations. We handled different notions for an optimal solution: the solution with minimum change or the solution in which the axioms are the most original axioms of the ontology. One of the major problems in ontology evolution process is loss of intended consequences, that is, as a result of ontology evolution we might lose some important consequences from the ontology. But, in context-based ontology evolution approaches as all the possible solutions are stored in iterative ontology update, if the user wants a specific consequence then, it is possible to choose the solution which entails the consequence as the optimal solution of the ontology update. Thus, loss of intended consequences can be minimized using this approach. We also developed methods to do context-based reasoning over the extracted optimal solutions.

We developed first prototypical implementation of all the theoretical methods developed for context-based ontology evolution operations. We developed a protége plug-in to provide a interface for the user to do context-based ontology evolution operations and context-based reasoning. We performed experiments on real-world ontologies FULL-GALEN, SNOMED CT and showed that our implementation works well on real-world ontologies. Our results show that, iterative ontology update with minimum change can be achieved using context-based ontology evolution approaches in a reasonable amount of

time. The in-depth analysis of the results obtained from the experiments show that limitations of our tool are: it is memory expensive, as all the approaches are implemented in Java which has bad garbage collection and which is memory expensive.

Future Research

There are several open avenues for future work to consider. Some of them can be listed as follows. To extract the optimal solutions from the labelled ontology that is obtained as result of context-based ontology evolution operations, we considered three notions for the Best. But, many notions for the best can be considered like: weights can be added to every axiom of the ontology, depending on the importance of the axioms and the optimal solution can be defined as the context with highest weight. This gives the solution which has the most important axioms of the ontology.

Downside of the algorithm to compute repairs to a consequence w.r.t every context in the boundary is that, it requires high memory to store the axioms and labels of all the axioms that belong to every repair computed. This algorithm can be improved by finding a better way to update the labels of the axioms in the ontology.

In empirical evaluation, we considered two types of cases for experiments: cases in which, only one optimal solution is considered after each context-based ontology evolution step and the cases in which, all the context-based ontology evolution steps are done on the whole ontology and only one optimal solution is selected at the end of all the steps. Instead, we can choose all the optimal solutions at the end of every step and do further updates on them. This method consumes relatively less time and memory than doing operations on the whole ontology in every step. This method can be checked if it controls the information loss during iterative ontology update.

In this thesis, we used a simple labelling function which assigns a natural number as a label to every axiom in the ontology. The labelling function can be improved to handle labels of the axioms efficiently.

We considered subsumption relation between atomic concepts as *consequence* in this thesis. It can be further developed to handle complex concept descriptions in addition to the atomic concepts.

Bibliography

[1] Calvanese D. McGuinness D.L. Nardi D. Petel-Schneider P.F. Baader, F. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2003.

[2] Patel-Schneider P.F. van Harmelen F. Horrocks, I. From SHIQ and RDF to OWL: the making of a web ontology language. *Journal of Web Semantics*, pages 7–26, 2003.

[3] Evgeny Kharlamov Bernardo Cuenca Grau and Dmitriy Zheleznyakov. Ontology contraction: Beyond the propositional paradise. *6th Alberto Mendelzon International Workshop on Foundations of Data Management (AMW2012), Ouro Preto, Brazil*, Vol-866, June 27-30, 2012.

[4] B. Cuenca Grau, E. Jimenez-ruiz, E. Kharlamov, and D. Zheleznyakov. Ontology evolution under semantic constraints.

[5] Gardenfors P. Makinson D Alchourr on, C.E. On the logic of theory change: Partial meet contraction and revision functions. *J. Symb. Log.*, 50(2):510–530, 1985.

[6] Sven Ove Hansson. A textbook of belief dynamics. *Applied Logic Series. Kluwer Academic Publishers*, 1999.

[7] Bernardo Cuenca Grau; Evgeny Kharlamov; and Dmitriy Zheleznyakov. How to contract ontologies. *OWLED Workshop*, vol-849, 2012.

[8] Parsia B. Sirin E. Hendler J.A Kalyanpur, A. Debugging unsatisfiable classes in OWL ontologies. *In Journal of Web Semantics*, 3(4):268–293, 2005.

[9] Huang Z. Cornet R. van Harmelen F. Schlobach, S. Debugging incoherent terminologies. *Journal of Automated Reasoning*, 39(3):317–349, 2007.

[10] Franz Baader, Martin Knechtel, and Rafael Peñaloza. Context-dependent views to axioms and consequences of semantic web ontologies. *Web Semant.*, 12-13:22–40, April 2012.

[11] Aparna Saisree Thuluva. Context labelling and context-based reasoning on semantic web ontologies. Project report.

[12] Aditya Kalyanpur, Bijan Parsia, Matthew Horridge, and Evren Sirin. Finding all justifications of OWL DL.

[13] B. Suntisrivaraporn. *Polynomial-time reasoning support for design and maintenance of large-scale biomedical ontologies*. PhD thesis, Technische Universitat Dresden, 2008.

[14] R Reiter. A theory of diagnosis from first principles. *Artif. Intell.*, 32(1):57–95, April 1987.

[15] Yu Sun and Chinese Academy of Sciences China Yuefei Sui, Key Laboratory of Intelligent Information Processing Institute of Computing Technology. The ontology revision.

[16] Michele Vescovi Roberto Sebastiani. Debugging large EL+ ontologies via SAT and SMT techniques. *Journal of Artificial Intelligence Research*, 1:1–80, 2010.

[17] Michel Ludwig. Just: a tool for computing justifications w.r.t. ELH ontologies. *OWL Reasoner Evaluation Workshop*, Vol- 1207, 2014.