unibz
Freie Universität Bozen
Libera Università di Bolzano
Università Liedia de Bulsan

Fakultät für Ingenieurwesen
Facoltà di Ingegneria
Faculty of Engineering

Bachelor Thesis

Knowledge Refinement in Expressive Description Logics

Candidate Roland Bernard

Supervisors Oliver Kutz
Nicolas Troquard

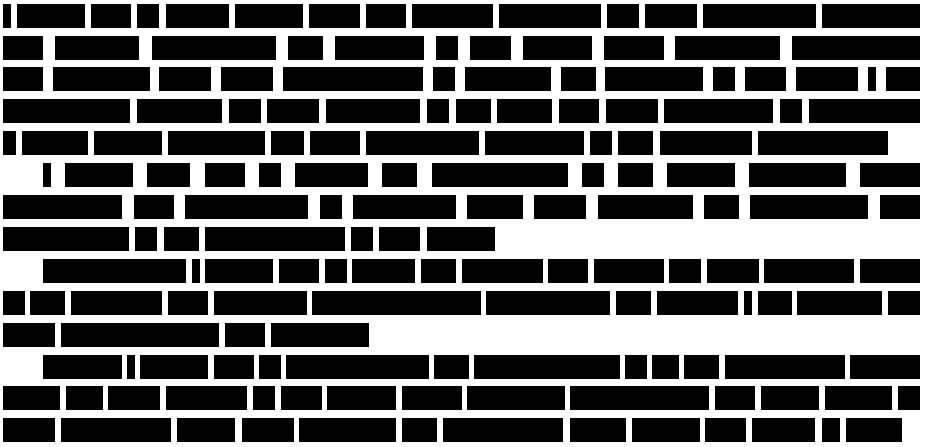
July 2023

Abstract

The field of ontology engineering plays a crucial role in knowledge representation and has gained significant attention in recent years in many domains, such as medicine and biology. The introduction of the Web Ontology Language as a W3C recommendation has further enabled use cases for ontology engineering in the context of the semantic web. With the growing size and complexity of these ontologies, however, they become more susceptible to bugs, and it becomes harder to debug defects. Moreover, in the context of the semantic web, automatic approaches to debugging ontologies are required for the combination of knowledge derived from conflicting sources. Axiom weakening is a technique that allows for the fine-grained repair of inconsistent ontologies. Its main advantage is that, through the use of refinement operators, it repairs ontologies by making axioms less restrictive rather than by removing them. Building on previously introduced axiom weakening for $\mathcal{ALC}$, this thesis presents an extension to the axiom weakening operator to deal with $\mathcal{SROIQ}$, the expressive and decidable description logic underlying OWL 2 DL. The main problem here is to ensure that the global constraints of $\mathcal{SROIQ}$ are preserved in the weakening process, as not every weaker axiom can be inserted into an ontology without compromising them. A basic regularity-preserving weakening approach for $\mathcal{SROIQ}$ is presented, and some alternatives are discussed. Further, the thesis briefly describes a prototype implementation and presents the results of basic experimental evaluations. Additionally, the implementation of the presented axiom weakening techniques as a plugin for the popular ontology editor, Protégé, is discussed.

Contents

Acknowledgements



Chapter 1

Introduction

The field of ontologies plays a crucial role in knowledge representation and has gained significant attention in recent years in many domains, such as biology and medicine. The introduction of the Web Ontology Language as a World Wide Web Consortium (W3C) recommendation has further grown the ecosystem and enabled use cases for ontology engineering in the context of the semantic web.

Ontologies provide a structured and formalized way to represent knowledge about particular domains and facilitate automatic reasoning and sharing knowledge. Nevertheless, the development and maintenance of ontologies is not without challenges. With the growing size and complexity of these ontologies, they also become more susceptible to bugs, and it becomes harder to debug these defects. Furthermore, in the context of the semantic web, automatic approaches to correcting ontologies are needed for the combination of conflicting knowledge derived from independent sources.

The $\mathcal{SROIQ}$ description logic is a very expressive variant of the description logics family, which can serve as the logical basis for representing knowledge in ontologies. It is also the foundation for the Web Ontology Language, a standardized language for representing ontologies on the web, enabling the creation of machine-readable ontologies. In both of these systems, ontologies are sets of axioms, and each axiom is a statement encoding some knowledge about the domain. $\mathcal{SROIQ}$ extends upon basic description logics like $\mathcal{ALC}$ with additional features such as complex role hierarchies, role disjointness axioms, and transitive roles. This expressive power allows for modelling complex domains with rich relationships and constraints. This expressivity, however, comes at the cost of requiring some global restrictions that must be followed to guarantee the decidability of the logic.

Many approaches to repairing inconsistent ontologies amount to identifying problematic axioms and then removing them (e.g., [58, 35, 34, 7]). While this approach is obviously sufficient to ensure that the resulting ontology is consistent, it tends to cause information loss as a secondary effect, as outlined in detail in related works [64, 16]. Axiom weakening has been proposed as a solution for fine-grained repair to inconsistent ontologies. In these approaches, the information loss is reduced by weakening the inferential power of an axiom rather than by deleting it entirely [19, 14, 6, 64, 16]. In [64], axiom weakening using refinement operators has been described for $\mathcal{ALC}$ and experimentally evaluated, showing that axiom weakening is able to retain more information than deletion. In [16], axiom weakening is extended to include many aspects of $\mathcal{SROIQ}$, notably omitting, however, the weakening of RBox axioms.

This thesis aims to explore the topic of ontology repair, focusing specifically on axiom weakening in the $\mathcal{SROIQ}$ description logic and the Web Ontology Language. It extends upon the previous work on axiom weakening in description logics by extending the underlying basic principles to the logic $\mathcal{SROIQ}$, including the possibility of weakening role inclusion axioms and disjoint role axioms. The thesis examines the challenges and difficulties that arise when applying axiom weakening to these more expressive description logics and covers several scenarios where weakening can affect the regularity of $\mathcal{SROIQ}$ RBoxes. Further, a framework where these problems can be prevented is proposed.

The implementation of the proposed algorithms will be discussed. Next to a prototype implementation used primarily for running the experimental evaluation, a plugin for the popular ontology editor, Protégé, has been realized. The thesis presents some details of the implementation process and discusses the integration of axiom weakening functionalities into the Protégé tool.

Additionally, by implementing the proposed refinement and weakening operators, it becomes possible to perform experimental evaluation, also on ontologies using the more expressive features of $\mathcal{SROIQ}$. The quality of repairs achieved through axiom weakening is assessed, and the results reaffirm the findings of [64] for the case of $\mathcal{SROIQ}$, namely that weakening may significantly outperform deletion. Additionally, the thesis explores the impact of caching in cover computations and evaluates the performance in terms of reasoner calls and execution time.

Finally, the thesis concludes with a summary of the findings, their implications, and possible directions for future research. By addressing some of the challenges related to axiom weakening in the $\mathcal{SROIQ}$ description logic, this research aims to contribute to the field of ontology engineering and inspire further research into axiom weakening and ontology repair.

Chapter 2

Background and Related Work

2.1 The $\mathcal{SROIQ}$ Description Logic

Formally, an ontology is a set of statements expressed in an appropriate logical language. The purpose of ontologies is the formal description of a specific domain of interest. While ontologies can be represented using several different formalisms, for use in automated reasoning a trade-off must be made between expressivity and practicality. For example, *first-order logic* (FOL) is more expressive than propositional logic, but this added expressivity comes at the cost of decidability. In addition to decidability, scalability must also be considered in the choice and design of the used representation.

Description logics (DL) are often used for building ontologies. They include a family of related knowledge representation languages and are often fragments of FOL[1] with equality, as is the case with the DL $\mathcal{SROIQ}$ which is the main focus of this work. DLs are almost always designed to be decidable and generally offer a favourable trade-off between expressivity and complexity of reasoning tasks. Different DLs have been developed for different applications and allow for varying levels of expressivity.

This section will briefly introduce the DL $\mathcal{SROIQ}$ [27, 56, 5]. A more detailed description, including more explanations and examples, can be found in Appendix A.

2.1.1 The Syntax of $\mathcal{SROIQ}$

The syntax of $\mathcal{SROIQ}$ is based on a *vocabulary* of three finite disjoint sets N_C, N_R, N_I of, respectively, *concept names*, *role names*, and *individual names*. The sets of, respectively, $\mathcal{SROIQ}$ *roles* and $\mathcal{SROIQ}$ *concepts* are generated from the following grammar.

$$
\begin{aligned}
R, S &::= U \mid r \mid r^{-} \ , \\
C &::= \bot \mid \top \mid A \mid \neg C \mid C \sqcap C \mid C \sqcup C \mid \forall R.C \mid \exists R.C \mid \\
&\qquad \geq n\ S.C \mid \leq n\ S.C \mid \exists S.\mathit{Self} \mid \{a\} \ ,
\end{aligned}
$$

where $A \in N_C$ is a concept name, $r \in N_R$ is a role name, $a \in N_I$ is an individual name and $n \in \mathbb{N}_0$ is a non-negative integer. U is the universal role. S is a *simple role* in the RBox $\mathcal{R}$ (see below). In the following, $\mathcal{L}(N_C, N_R, N_I)$ and $\mathcal{L}(N_R) =$

[1] While DLs are fragments of FOL in the sense that for every ontology in a given DL, there exists a FOL theory that has the same models, the syntax used by DLs is different from the syntax used in FOL, and as such a DL axiom is not a valid FOL sentence.

$N_R \cup \{U\} \cup \{r^- \mid r \in N_R\}$ denote, respectively, the set of concepts and roles that can be built over N_C, N_R, and N_I in $\mathcal{SROIQ}$.

We next define the notions of TBox, ABox, and (regular) RBox, of complex role inclusions, and of (non-)simple roles: A *TBox* $\mathcal{T}$ is a finite set of *concept inclusions* (GCIs) of the form $C \sqsubseteq D$ where C and D are concepts. The TBox is used to store terminological knowledge concerning the relationships between concepts. An *ABox* $\mathcal{A}$ is a finite set of statements of the form $C(a)$, $R(a,b)$, $\neg R(a,b)$, $a = b$, and $a \neq b$, where C is a concept, R is a role and a and b are individual names. The ABox expresses knowledge regarding individuals in the domain. An *RBox* $\mathcal{R}$ is a finite set of *role inclusions* (RIAs) of the form $R_1 \circ \cdots \circ R_n \sqsubseteq R$, and disjoint role axioms $\mathrm{disjoint}(S_1, S_2)$ where R, R_1, ..., R_n, S_1, and S_2 are roles. S_1 and S_2 are simple (defined next) in the RBox $\mathcal{R}$. The special case of $n = 1$ is a *simple role inclusion*, while we call the cases where $n > 1$ *complex role inclusions*. The RBox represents knowledge about the relationships between roles.

The set of *non-simple* roles in $\mathcal{R}$ is the smallest set such that: U is non-simple; any role R that appears as the super role of a complex RIA $R_1 \circ \cdots \circ R_n \sqsubseteq R$ where $n > 1$ is non-simple; any role R that appears on the right-hand side of a simple RIA $S \sqsubseteq R$ where S is non-simple, is also non-simple; and a role r is non-simple if and only if r^- is non-simple. All other roles are *simple*.

For convenience, let us define the function $\mathrm{inv}(R)$ such that $\mathrm{inv}(U) = U$, $\mathrm{inv}(r) = r^-$, and $\mathrm{inv}(r^-) = r$ for all role names $r \in N_R$. An RBox $\mathcal{R}$ is *regular* if there exists a preorder $\preceq$, i.e., a transitive and reflexive relation, over the set of roles appearing in $\mathcal{R}$, such that $R \preceq S \iff \mathrm{inv}(R) \preceq S$ and all RIAs in $\mathcal{R}$ are of the forms: $\mathrm{inv}(R) \sqsubseteq R$, $R \circ R \sqsubseteq R$, $S \sqsubseteq R$, $R \circ S_1 \circ \cdots \circ S_n \sqsubseteq R$, $S_1 \circ \cdots \circ S_n \circ R \sqsubseteq R$, or $S_1 \circ \cdots \circ S_n \sqsubseteq R$, where $n > 1$ and R, S, $S_1, \cdots, S_n$ are roles such that $S \preceq R$, $S_i \preceq R$, and $R \not\preceq S_i$ for $i = 1, \ldots, n$. This regularity restriction has been chosen to align with the implementation of the OWL 2 DL [50] profile checker in the OWL API [24].

Definition 2.1. A $\mathcal{SROIQ}$ ontology $\mathcal{O} = \mathcal{T} \cup \mathcal{A} \cup \mathcal{R}$ consists of a TBox $\mathcal{T}$, an ABox $\mathcal{A}$, and an RBox $\mathcal{R}$ in the language of $\mathcal{SROIQ}$, and where the RBox $\mathcal{R}$ is regular.

2.1.2 The Semantics of $\mathcal{SROIQ}$

The semantics of $\mathcal{SROIQ}$ is defined using *interpretations* $I = \langle \Delta^I, \cdot^I \rangle$, where Δ^I is a non-empty *domain* and $\cdot^I$ is a function associating to each individual name a an element of the domain $a^I \in \Delta^I$, to each concept C a subset of the domain $C^I \subseteq \Delta^I$, and to each role R a binary relation on the domain $R^I \subseteq \Delta^I \times \Delta^I$. The interpretation function $\cdot^I$ is extended to the interpretation of complex roles and concepts, such that

$$
\begin{aligned}
U^I = \Delta^I \times \Delta^I \ , \qquad \top^I = \Delta^I \ , \qquad \bot^I = \emptyset \ , \qquad \{a\}^I = \{a^I\} \ , \\
(\neg C)^I = \Delta^I \setminus C^I \ , \quad (C \sqcap D)^I = C^I \cap D^I \ , \quad (C \sqcup D)^I = C^I \cup D^I \ , \\
(r^-)^I = \{\langle \delta_2, \delta_1 \rangle \mid \langle \delta_1, \delta_2 \rangle \in r^I\} \ , \\
(\forall R.C)^I = \{\delta \in \Delta^I \mid \forall \delta' \in \Delta^I : \langle \delta, \delta' \rangle \in R^I \implies \delta' \in C^I\} \ , \\
(\exists R.C)^I = \{\delta \in \Delta^I \mid \exists \delta' \in \Delta^I : \langle \delta, \delta' \rangle \in R^I \text{ and } \delta' \in C^I\} \ , \\
(\leq n\ R.C)^I = \{\delta \in \Delta^I \mid \left|\{\delta' \in \Delta^I \mid \langle \delta, \delta' \rangle \in R^I \text{ and } \delta' \in C^I\}\right| \leq n\} \ , \\
(\geq n\ R.C)^I = \{\delta \in \Delta^I \mid \left|\{\delta' \in \Delta^I \mid \langle \delta, \delta' \rangle \in R^I \text{ and } \delta' \in C^I\}\right| \geq n\} \ , \\
(\exists R.\mathit{Self})^I = \{\delta \in \Delta^I \mid \langle \delta, \delta \rangle \in R^I\} \ ,
\end{aligned}
$$

where $|X|$ denotes the cardinality of a set X. An interpretation I is a *model for* $\mathcal{O}$, written $I \models \mathcal{O}$, if it satisfies all the axioms in $\mathcal{O}$. Whether the interpretation I satisfies the axiom α, written $I \models \alpha$, is given by

$$
\begin{aligned}
I \models C \sqsubseteq D &\iff C^I \subseteq D^I \ , & I \models R(a,b) &\iff \langle a^I, b^I \rangle \in R^I \ , \\
I \models C(a) &\iff a^I \in C^I \ , & I \models \neg R(a,b) &\iff \langle a^I, b^I \rangle \notin R^I \ , \\
I \models a = b &\iff a^I = b^I \ , & I \models \mathrm{disjoint}(R,S) &\iff R^I \cap S^I = \emptyset \ , \\
I \models a \neq b &\iff a^I \neq b^I \ , & I \models S_1 \circ \cdots \circ S_n \sqsubseteq R &\iff S_1^I \circ \cdots \circ S_n^I \subseteq R^I \ ,
\end{aligned}
$$

where $\circ$ denotes the composition of relations, that is, $r \circ s = \{\langle x, y \rangle \mid \exists z : \langle x, z \rangle \in r \text{ and } \langle z, y \rangle \in s\}$.

An ontology $\mathcal{O}$ is *consistent* if there exists a model $I \models \mathcal{O}$ for the ontology. An ontology that is not consistent is *inconsistent*. A concept C is *satisfiable* in an ontology $\mathcal{O}$ if there exists a model $I \models \mathcal{O}$ in which $C^I \neq \emptyset$. An ontology in which all atomic concepts $C \in N_C$ are satisfiable is called *coherent*. We say an axiom α is *entailed by* the ontology $\mathcal{O}$, written $\mathcal{O} \models \alpha$, if and only if every model $I \models \mathcal{O}$ of the ontology satisfies $I \models \alpha$.

Given two concepts C and D we say that C is *subsumed* by D (or D *subsumes* C) with respect to the ontology $\mathcal{O}$, written $C \sqsubseteq_{\mathcal{O}} D$, if $C^I \subseteq D^I$ in every model I of $\mathcal{O}$. Further, C is *strictly subsumed by* D, written $C \sqsubset_{\mathcal{O}} D$, if $C \sqsubseteq_{\mathcal{O}} D$ but not $D \sqsubseteq_{\mathcal{O}} C$. Analogously, given two roles R and S, R is subsumed by S with respect to $\mathcal{O}$ (written $R \sqsubseteq_{\mathcal{O}} S$) if $R^I \sqsubseteq S^I$ in all models I of $\mathcal{O}$. Again, $R \sqsubset_{\mathcal{O}} S$ holds if $R \sqsubseteq_{\mathcal{O}} S$ but not $S \sqsubseteq_{\mathcal{O}} R$.

2.1.3 The $\mathcal{ALC}$ Description Logic

Since $\mathcal{ALC}$ will be used for part of the discussion on related work, this is a brief section discussing what features are included in $\mathcal{ALC}$, and which are not. $\mathcal{ALC}$ is a much less expressive logic compared to $\mathcal{SROIQ}$. The vocabulary in $\mathcal{ALC}$ consists of the same components N_C, N_R, and N_I as in $\mathcal{SROIQ}$. In $\mathcal{ALC}$, concept expressions are formed only using the following grammar.

$$C ::= \bot \mid \top \mid A \mid \neg C \mid C \sqcap C \mid C \sqcup C \mid \forall R.C \mid \exists R.C \ ,$$

where $A \in N_C$ is a concept name and $r \in N_R$ is a role name. The universal role or inverse roles are not part of $\mathcal{ALC}$. Further, $\mathcal{ALC}$ supports only two types of axioms, GCIs of the form $C \sqsubseteq D$ and class assertions $C(a)$ where C and D are concepts and $a \in N_I$ is an individual name. It should be noted that $\mathcal{ALC}$ has the unique name assumption, meaning that two different individual names must necessarily refer to different elements of the domain in every interpretation. Aside from that, the axioms and concepts have the same semantic meaning as their direct counterparts in $\mathcal{SROIQ}$.

2.2 The Web Ontology Language

Description logics are also the basis of the *Web Ontology Language* (OWL) [23, 50], which is a World Wide Web Consortium (W3C) recommendation and is intended for use as part of the semantic web. While the OWL 2 DL language is based on $\mathcal{SROIQ}$, OWL 2 also defines three so-called profiles that are fragments of the full OWL 2 language and trade-off expressive power for more efficient reasoning [48, 50]. OWL 2

EL is a subset based on the DL $\mathcal{EL}^{++}$ and allows the basic reasoning task to be performed in polynomial time. This fragment is useful for ontologies with many classes and properties. OWL 2 QL is designed for applications with many instances where query answering is an important task. OWL 2 RL is designed such that reasoning tasks can be implemented using a rule-based reasoning engine. The OWL 2 DL profile, which is the most expressive of the profiles that is still decidable[2], is based on $\mathcal{SROIQ}$. This thesis, therefore, focuses on OWL 2 DL.

It is important to note that while OWL 2 is based on DLs like $\mathcal{SROIQ}$, it provides many more axiom and concept expression types than are natively available in $\mathcal{SROIQ}$. It provides, for example, axioms that allow specifying disjointness between concepts, or ones that allow the description of equivalent roles. These additional axioms do, however, not make the language more expressive, as all of them can equivalently be expressed using one or more $\mathcal{SROIQ}$ axioms. In contrast, OWL 2 DL also provides some features that can not be reproduced in pure $\mathcal{SROIQ}$, e.g., annotations, data types and data properties. For the rest of this thesis, these will not be considered, and they have been removed from the ontologies used in the evaluation, as will be explained in further detail in Chapter 5.

In addition to the different profiles, there exist also different syntaxes for OWL 2. The syntax defined and used by the main specification [50] is the so-called functional syntax. An alternative ontology format supporting OWL 2 is the XML/RDF-based format [9, 49], which is the main format used for exchanging ontologies between different tools. It is the only format whose support is mandated for compliant tools by the specification. The XML version of the RDF format is not designed to be presented to and manipulated directly by human users, but is suitable for machine processing and interoperability. The RDF format can also be used for other forms of data and can be written in an alternative non-XML syntax called Turtle [10] that is more human-readable. Another popular format for OWL 2 is the Manchester syntax [26], which is also the default used in the popular ontology development tool Protégé (see below).

Highly optimized reasoners exist for different OWL profiles. Reasoners are programs that, given an ontology, can perform certain reasoning tasks like checking for consistency of the ontology and testing whether axioms are entailed or not. Some reasoners specialize in the more restricted profiles, e.g., ELK [39] supports only the OWL 2 EL profile. Others, like the one used for this thesis, have complete or almost complete support for OWL 2 DL, and therefore also $\mathcal{SROIQ}$. For this thesis, as will be explained in more detail in Chapter 4, we are using the reasoners through the OWL API [24]. The OWL API is a Java library that provides data structures and utilities to represent OWL 2 ontologies, load and store them to disk, and perform several basic transformations. Further, it provides a common way of interfacing with different reasoners to query consistency, entailment, and class hierarchies.

Protégé [51] is a popular tool for ontology development built on top of the OWL API. It allows reading, writing, viewing, and modifying ontologies in all main formats supporting OWL 2. As mentioned above, it uses the Manchester syntax for displaying ontology axioms and has support for plugins. These can be used to add different functionalities like reasoners, debuggers, or ontology analysis tools. In Section 4.2, we show the implementation of a Protégé plugin supporting the application of the algorithms proposed in this thesis to the ontologies loaded in the editor.

[2]The most expressive profile, OWL 2 Full, is not decidable.

2.3 Ontology Bugs

As software systems evolve, it becomes harder to avoid introducing bugs. Similarly, in ontology engineering, bugs can be introduced into an ontology. With the increasing size and complexity of a system, it becomes harder to debug these defects, both for software systems and ontologies.

2.3.1 Categories of Bugs

Defects, in both software systems and ontologies, can be due to several different reasons. In [35], the authors identify three broad categories of defects that can be present in an ontology: *syntactic defects*, *semantic defects,* and *modelling defects*.

Syntactic Defects

Syntactic defects in an ontology can be caused by a statement that does not conform to the grammar of the employed logic. These sorts of defects are easy to locate and correct. In general, tool support for these kinds of defects is able to pinpoint the location of the defect and give an explanation to the user.

Example 2.1. The axioms $C \sqsubseteq\sqsubseteq D$, $C(r(a,b))$, or $C \sqsubset D$ are all obviously not syntactically correct $\mathcal{SROIQ}$ axioms. Similarly, the concepts $C \sqcup \sqcap D$, $\leq 3\ t.Self$, and $Self \sqcup D$ do not match the grammar for concept expressions.

There may, however, be some additional restrictions on what constitutes a valid ontology that are not based solely on the grammatical rules. For ontologies, these might be, for example, the restrictions placed upon the RDF graph for a specific OWL profile. In the case of $\mathcal{SROIQ}$, these include the restricted use of non-simple roles and the regularity condition placed on the RBox. Restrictions of this kind can often be much easier to violate and harder to debug than the syntactic defects which are simple violations of the grammar.

Example 2.2. Given the vocabulary $N_C = \{C\}$ and $N_R = \{a, b, c\}$, the $\mathcal{SROIQ}$ ontology $\mathcal{O} = \{a \sqsubseteq b, b \circ c \sqsubseteq a\}$ is not valid. This is because it does not satisfy the regularity condition in $\mathcal{SROIQ}$. This can easily be verified by showing that no preorder $\preceq$ exists for which $a \preceq b$ and $a \not\preceq b$ hold.

Similarly, the ontology $\mathcal{O} = \{a \circ a \sqsubseteq a, \top \sqsubseteq \exists a.Self\}$ is invalid because the role a is non-simple. Since roles used in $Self$ constraints must be simple, the second axiom is not allowed.

Semantic Defects

For ontologies, semantic defects, as defined in [35], are those which can be discovered by a reasoner given an ontology free of syntactic defects. This includes, for example, the inconsistency of the ontology or the unsatisfiability of a concept. The presence of such defects is generally not hard to identify, given the availability of a reasoner for the logic of the ontology. It is, however, often not trivial to understand the underlying source of the defect. In this thesis, the focus will be mainly on these kinds of bugs, especially on the problem of inconsistent ontologies.

Example 2.3. Given the vocabulary $N_C = \{C, D\}$ and $N_I = \{a\}$, the $\mathcal{SROIQ}$ ontology $\mathcal{O} = \{D \sqsubseteq \neg C, D(a), C(a)\}$ is inconsistent. This can be seen clearly from

the fact that for every interpretation I, a^I must be in D^I, but this means that a^I must not be in C^I, which contradicts the last axiom.

Modelling Defects

Modelling defects are all those defects that are not syntactic or semantic defects. The presence of unintended or absence of intended inferences is one such defect. These defects can also be of a more stylistic nature. Redundancy or unused parts of the ontology may be considered defects, since they do not add any knowledge to the ontology.

These kinds of defects can in general not be detected automatically by tools. They require careful attention and domain-specific knowledge to be revealed and corrected. In some scenarios, testing may be used to uncover and prevent some modelling defects by expressing more explicitly the intention of the modeller.

2.3.2 Causes of Bugs

We will now briefly explore the possible causes of bugs in the context of ontology engineering. One major source of defects is of course human error by the ontology developer. These can range from trivial typos, that will most often merely result in syntactic errors, to more fundamental misunderstandings of the domain, which can lead to modelling mistakes. Especially syntactic errors concerning the additional constraints, like regularity and simplicity, can easily be violated due to an oversight by the modeller. One can, for example, define a role as transitive and later use it in cardinality constraints. Unfortunately, however, this is not allowed in many ontology languages, such as in $\mathcal{SROIQ}$ or OWL 2 DL. Similarly, mistakes during the modelling process can cause inconsistent ontologies or unsatisfiable concepts. The modeller might have defined two classes as disjoint, but later added an individual belonging to the intersection of the two concepts, yielding an inconsistent ontology.

Another common source for bugs in ontology engineering arises when merging different ontologies. This may be done, especially in the context of OWL and the semantic web, to combine the knowledge obtained from different sources. This can again be a case where one ontology defines a certain role as transitive, while the other one assumes that it is simple. Combining the two ontologies will then obviously lead to a violation of the global constraints. Also, different ontologies might make different but incompatible modelling choices when modelling the same domain. This can result in an ontology after merging that is inconsistent or has unsatisfiable concepts.

Example 2.4. Given the vocabulary $N_C = \{C, D\}$ and $N_I = \{a, b\}$, the two $\mathcal{SROIQ}$ ontology $\mathcal{O}_1 = \{D \sqsubseteq C, D(a)\}$ and $\mathcal{O}_2 = \{C \sqsubseteq \neg D, C(b)\}$ are by themselves consistent. If we merge them, however, the resulting ontology $\mathcal{O}_1 \cup \mathcal{O}_2$ is no longer consistent.

2.4 Repairing Ontologies

As established in Section 2.3, maintaining the consistency and correctness of ontologies can be a difficult task. Ontology repair is the process of automatically correcting these inconsistencies or errors in ontologies. Several approaches have been proposed for ontology repair. This section will explore some of these approaches and their underlying principles.

2.4.1 Basic Definitions

We define a repair as proposed in [6]. It is assumed, as is the case for most DLs, that there exists a monotone consequence operator $\models$ such that for two ontologies $\mathcal{O}_1 \subseteq \mathcal{O}_2$ and axiom α, $\mathcal{O}_1 \models \alpha$ implies $\mathcal{O}_2 \models \alpha$. Additionally, $\mathrm{Con}(\mathcal{O})$ shall contain all consequences of $\mathcal{O}$, that is $\mathrm{Con}(\mathcal{O}) = \{\alpha \mid \mathcal{O} \models \alpha\}$. We split the ontology further into two disjoint sets $\mathcal{O} = \mathcal{O}_s \cup \mathcal{O}_r$ of *static axioms* $\mathcal{O}_s$ and *refutable axioms* $\mathcal{O}_r$. Static axioms are assumed to be correct and may not be touched by the repair procedure, while refutable axioms are possibly erroneous. This separation is useful, for example, if the static part of the ontology is hand-crafted, while the refutable part has been automatically generated. Similarly, it is applicable in case multiple ontologies are combined, and some sources are seen as less trustworthy than others.

Definition 2.2. Given an ontology $\mathcal{O} = \mathcal{O}_s \cup \mathcal{O}_r$ and an unintended consequence $\mathcal{O} \models \alpha$, $\mathcal{O}_s \not\models \alpha$, the ontology $\mathcal{O}_s \subseteq \mathcal{O}'$ is a *repair* of $\mathcal{O}$ with respect to α if $\mathrm{Con}(\mathcal{O}') \subseteq \mathrm{Con}(\mathcal{O}) \setminus \{\alpha\}$. A repair $\mathcal{O}'$ is an *optimal repair* of $\mathcal{O}$ with respect to α if there exists no other repair $\mathcal{O}_s \subseteq \mathcal{O}''$ such that $\mathrm{Con}(\mathcal{O}') \subset \mathrm{Con}(\mathcal{O}'') \subseteq \mathrm{Con}(\mathcal{O}) \setminus \{\alpha\}$.

Given that $\mathcal{O}_s \not\models \alpha$, a repair is guaranteed to exist, since $\mathcal{O}_s$ is one such repair. On the other hand, as has been shown in [6], generally, an optimal repair does not necessarily need to exist.

Example 2.5. Given the vocabulary $N_C = \{A\}$ and $N_I = \{a\}$, let the ontology $\mathcal{O} = \mathcal{O}_s \cup \mathcal{O}_r$ be made up of the static axioms $\mathcal{O}_s = \{A \sqsubseteq \exists r.A, \exists r.A \sqsubseteq A\}$ and refutable axioms $\mathcal{O}_r = \{A(a)\}$. Let $\alpha = A(a)$ be an unintended consequence of $\mathcal{O}$. In this case, the ontology $\mathcal{O}' = \mathcal{O}_s \cup \{(\exists r.\top)(a)\}$ is a possible repair. However, using any axioms $((\exists r.)^n\top)(a)$ also yields a valid repair. Since for every repair using $((\exists r.)^n\top)(a)$, there exists a repair using $((\exists r.)^{n+1}\top)(a)$ that has more consequences, there exists no optimal repair.

It should be noted also that there exists an infinite number of possible repairs, as adding tautologies to a repair will always yield another valid repair. In the case that we are interested in making an inconsistent ontology consistent, we can use as α any unsatisfiable axiom, e.g., $\top \sqsubseteq \bot$. Since all axioms, including unsatisfiable axioms, are entailed by inconsistent ontologies, a repair that does not entail α is consistent. Notice also that in this case where $\mathcal{O}$ is inconsistent, any consistent ontology that does not entail α, even if completely unrelated to $\mathcal{O}$, will be a repair of $\mathcal{O}$. It will be assumed in the rest of this thesis that unless otherwise noted all axioms of an ontology are refutable, and that the unintended consequence is the inconsistency of the ontology.

In contrast, the classical approach to repair consists of identifying and removing problematic axioms (e.g, [58, 35, 34, 7]). As such, a classical repair is always a subset of the original ontology and the number of classical repairs for any pair $\mathcal{O}$ and α is necessarily finite.

Definition 2.3. Given an ontology $\mathcal{O} = \mathcal{O}_s \cup \mathcal{O}_r$ and an unintended consequence $\mathcal{O} \models \alpha$, $\mathcal{O}_s \not\models \alpha$, the ontology $\mathcal{O}_s \subseteq \mathcal{O}' \subseteq \mathcal{O}$ is a *classical repair* of $\mathcal{O}$ with respect to α if $\mathrm{Con}(\mathcal{O}') \subseteq \mathrm{Con}(\mathcal{O}) \setminus \{\alpha\}$. A classical repair $\mathcal{O}'$ is an *optimal classical repair* of $\mathcal{O}$ with respect to α if there exists no other classical repair $\mathcal{O}_s \subseteq \mathcal{O}'' \subseteq \mathcal{O}$ such that $\mathrm{Con}(\mathcal{O}') \subset \mathrm{Con}(\mathcal{O}'') \subseteq \mathrm{Con}(\mathcal{O}) \setminus \{\alpha\}$.

We can observe that every classical repair is, in fact, a valid repair, and that also a classical repair is guaranteed to exist. Unlike for the general case of optimal repairs,

an optimal classical repair is always guaranteed to exist. This follows from the fact that the set of classical repairs is finite, and the $\subset$ relation is a strict partial order, so there can not be an infinite sequence of classical repairs $\mathcal{O}^{(1)}, \mathcal{O}^{(2)}, \ldots$ such that $\mathrm{Con}(\mathcal{O}^{(i)}) \subset \mathrm{Con}(\mathcal{O}^{(i+1)})$.

2.4.2 Repair Approaches

Classical Repairs

Generating a classical repair can be achieved in a number of equivalent ways. One way to compute an optimal classical repair is using justifications and hitting sets [53].

Definition 2.4. Given an ontology $\mathcal{O} = \mathcal{O}_s \cup \mathcal{O}_r$ and an axiom $\mathcal{O} \models \alpha$, $\mathcal{O}_s \not\models \alpha$, a *justification* for α with respect to $\mathcal{O}$ is a minimal subset $J \subseteq \mathcal{O}_r$ such that $J \cup \mathcal{O}_s \models \alpha$. Given the set of all justifications $J_1, \ldots, J_n$ for α, a *hitting set* H for these justifications is a set of axioms such that $H \cap J_i \neq \emptyset$ for $i = 1, \ldots, n$. H is a *minimal hitting* set if it does not strictly contain another hitting set.

Example 2.6. Given the vocabulary $N_C = \{C, D\}$ and $N_I = \{a\}$, the ontology $\mathcal{O} = \{D \sqsubseteq \neg C, D(a), C(a)\}$ is inconsistent. A possible (optimal) classical repair to restore the consistency is $\mathcal{O}' = \{D(a), C(a)\}$.

Justifications are always non-empty since $\mathcal{O}_s \not\models \alpha$, and therefore hitting sets and minimal hitting sets necessarily exist. Given any minimal hitting set H for the justification $J_1, \ldots, J_n$ of α in $\mathcal{O}$, the ontology $\mathcal{O}' = \mathcal{O} \setminus H$ is an optimal classical repair of $\mathcal{O}$ with respect to α.

This algorithm for computing optimal classical repairs requires the computation of all justifications, which can in general be very computationally intensive. Black-box approaches for computing justifications have been proposed [33, 58, 59] that compute justifications by repeatedly making calls to pre-existing highly-optimized reasoners. To compute all justification, however, they may in the worst case, need to make an exponential number of calls to the reasoner. Nevertheless, in practice, they may often be fast enough. For example, the hitting-set-tree based algorithm presented in [33] can be used to compute both all justifications and all hitting sets. There exist also glass-box approaches to computing justifications [33], that require only a single reasoning request to find justifications, but they also require specialized, generally less efficient, reasoners.

An alternative to computing all justification is to directly find a minimal correction subset C of $\mathcal{O}_r$ such that $\mathcal{O} \setminus C \not\models \alpha$. Finding a single such set can be done efficiently using similar algorithms to the ones for finding single justifications. Algorithms for solving the minimal subset over monotone predicate problem, such as the QuickXplain algorithm [31] or a progression-based algorithm [46] may be used. A subset of all such sets can be found efficiently using the MergeXplain algorithm [61]. Of course, computing all minimal correction subsets directly is also possible, using similar algorithms to the ones used for computing all justifications [45].

More Gentle Repairs

While the classical approach is obviously sufficient to guarantee that a possible repair is found, it can lead to unnecessary information loss. That is, the repaired ontology might be missing some consequences of the original ontology that were actually desirable and did not necessarily have to be removed to find a repair.

Example 2.7. Given the inconsistent ontology $\mathcal{O} = \{\top \sqsubseteq C \sqcap D, C \sqcap D \sqsubseteq \bot\}$, we want to repair it such that it becomes consistent. To repair this ontology using a classical repair, we have to remove one of the two axioms. However, to make the ontology consistent, it would be sufficient to remove one of the disjuncts in the first axiom.

Since, ideally, one wants to retain as much information as possible, alternative methods for repairing ontologies have been proposed that are able to preserve more information than the classical approach (e.g., [19, 14, 6, 64, 16, 25, 42]).

One option is to first modify the original ontology and afterwards apply the classical repair approach. The intuition is that in the modified ontology the individual axioms should contain less information, and therefore, the removal of axioms lead to less information loss relative to doing the same in the unmodified ontology. In [25] the authors propose a structural transformation, that replaces axioms with a set of weaker axioms that are semantically equivalent, but on their own contain less information.

Example 2.8. Given the inconsistent ontology $\mathcal{O} = \{A \sqsubseteq B \sqcap \neg C, B \sqsubseteq C, A(a)\}$, we want to repair it such that it becomes consistent. To repair this ontology using a classical repair, we have to remove one of the three axioms. However, we can transform it into the equivalent ontology $\mathcal{O}' = \{A \sqsubseteq B, A \sqsubseteq \neg C, B \sqsubseteq C, A(a)\}$. Now, we have some additional possibilities. We could for example remove only the axiom $A \sqsubseteq \neg C$ and retain the consequence $C(a)$ which would otherwise have always been lost.

Another approach to repairing ontologies more gently, that has been proposed in the literature, is using *axiom weakening* [64, 16, 6, 42]. Instead of removing axioms, they are replaced with weaker axioms. Replacing an axiom with a weaker axiom can not cause new consequences, but may diminish the set of consequences.

Definition 2.5. An axiom α is *weaker* that another axioms α' with respect to some ontology $\mathcal{O}$ if and only if for every model $I \models \mathcal{O}$ of $\mathcal{O}$, $I \models \alpha$ implies $I \models \alpha'$. Equivalently, we write $\alpha \models_{\mathcal{O}} \alpha'$.

In [42] the authors show a method for pinpointing the causes for unsatisfiability within axioms and propose a way of weakening axioms guided by this information. The authors of [6] show general theoretical results for repair using axiom weakening, and propose a concrete weakening relation for the DL $\mathcal{EL}$. They further show that the repair algorithm using their proposed axiom weakening terminates in at most an exponential number of weakening steps. [64] presents the repair of inconsistent ontologies using axiom weakening with the help of refinement operators. It is further empirically shown in [64] that weakening axioms can retain significantly more information compared to removing them. This approach is later extended in [16] to cover more expressive DLs, including most concepts of $\mathcal{SROIQ}$. Additionally, in [16] it is shown that the proposed repair algorithm using axiom weakening will almost surely, i.e., with probability 1, terminate.

Whether optimal repairs exist for these axiom weakening based approaches depends largely on which method is used for selecting weaker axioms. Replacement of the axioms with only tautologies is one trivial form of axiom weakening, and equivalent to generating classical repairs. On the other hand, the axiom weakening based repair approaches studied in [64, 16] and the one used in this thesis do not necessarily guarantee the existence of an optimal repair. This can be seen from the fact that Example 2.5 is applicable also to those algorithms.

2.5 Axiom Weakening in $\mathcal{ALC}$

Axiom weakening, as discussed in this thesis, is based on the approach presented for weakening with $\mathcal{ALC}$ ontologies in [64]. To provide the necessary context for the following discussion on how weakening may be performed in $\mathcal{SROIQ}$, we provide a brief overview of axiom weakening in the case of $\mathcal{ALC}$. For further details, see [64]. We begin by defining subconcepts. We will immediately also extend the definition to $\mathcal{SROIQ}$ concepts, since we will need it for later definitions of axiom weakening in $\mathcal{SROIQ}$.

Definition 2.6. Let $\mathcal{O}$ be an $\mathcal{ALC}$ or $\mathcal{SROIQ}$ ontology. The set of *subconcepts* of $\mathcal{O}$ is given by

$$\mathrm{sub}(\mathcal{O}) = \{\top, \bot\} \cup \bigcup_{C(a)\in\mathcal{O}} \mathrm{sub}(C) \cup \bigcup_{C\sqsubseteq D\in\mathcal{O}} (\mathrm{sub}(C) \cup \mathrm{sub}(D)) \ ,$$

where $\mathrm{sub}(C)$ is the set of *subconcepts* in C given by

$$\begin{aligned}
\mathrm{sub}(A) &= \{A\} \quad , A \in N_C \cup \{\top, \bot\} \ , & \mathrm{sub}(\neg C) &= \{\neg C\} \cup \mathrm{sub}(C) \ , \\
\mathrm{sub}(C \sqcup D) &= \{C \sqcup D\} \cup \mathrm{sub}(C) \cup \mathrm{sub}(D) \ , & \mathrm{sub}(\forall R.C) &= \{\forall R.C\} \cup \mathrm{sub}(C) \ , \\
\mathrm{sub}(C \sqcap D) &= \{C \sqcap D\} \cup \mathrm{sub}(C) \cup \mathrm{sub}(D) \ , & \mathrm{sub}(\exists R.C) &= \{\exists R.C\} \cup \mathrm{sub}(C) \ , \\
\mathrm{sub}(\geq n\ R.C) &= \{\geq n\ R.C\} \cup \mathrm{sub}(C) \ , & \mathrm{sub}(\leq n\ R.C) &= \{\leq n\ R.C\} \cup \mathrm{sub}(C) \ , \\
\mathrm{sub}(\exists R.\mathit{Self}) &= \{\exists R.\mathit{Self}\} \ , & \mathrm{sub}(\{a\}) &= \{\{a\}\} \ .
\end{aligned}$$

Let us now define the so-called upward and downward cover sets of concepts. The upward cover for a given concept is the set of the most specific generalizations from the (fixed) set of subconcepts, while the downward cover set contains the most general specializations from the same set of subconcepts. (It should be noted that the related open search for, e.g., a least common subsumer is in general a harder problem [4].)

Definition 2.7. Let $\mathcal{O}$ be an $\mathcal{ALC}$ ontologies that uses the vocabulary N_C, N_R, and N_I. The *upward cover* and *downward cover* for a concept C are given by

$$\begin{aligned}
\mathrm{UpCover}_{\mathcal{O}}(C) = \{&D \in \mathrm{sub}(\mathcal{O}) \mid C \sqsubseteq_{\mathcal{O}} D \text{ and} \\
&\nexists D' \in \mathrm{sub}(\mathcal{O}) \text{ with } C \sqsubset_{\mathcal{O}} D' \sqsubset_{\mathcal{O}} D\} \ , \\
\mathrm{DownCover}_{\mathcal{O}}(C) = \{&D \in \mathrm{sub}(\mathcal{O}) \mid D \sqsubseteq_{\mathcal{O}} C \text{ and} \\
&\nexists D' \in \mathrm{sub}(\mathcal{O}) \text{ with } D \sqsubset_{\mathcal{O}} D' \sqsubset_{\mathcal{O}} C\} \ .
\end{aligned}$$

Note here that the upward and downward covers will only yield useful results if the ontology $\mathcal{O}$ is consistent. Since they operate only over the subconcepts of $\mathcal{O}$, on their own, the upward and downward covers of concepts are missing some interesting refinements.

Example 2.9. Let $N_C = \{A, B, C\}$, $N_R = \{r\}$, and $\mathcal{O} = \{A \sqsubseteq B\}$. $\mathrm{sub}(\mathcal{O}) = \{\top, \bot, A, B\}$. The upward cover of $C \sqcup A$ is equal to $\mathrm{UpCover}_{\mathcal{O}}(C \sqcup A) = \{\top\}$. The potential refinement to $C \sqcup B$ will be missed even by repeated application of the upward cover because $C \sqcup B \notin \mathrm{sub}(\mathcal{O})$. Similarly, $\mathrm{UpCover}_{\mathcal{O}}(\forall r.A) = \{\top\}$, even though $\forall r.B$ is a reasonable generalization.

To additionally capture these omissions, we define generalization and specialization operators that recursively exploit the complex structure of the concept expressions being refined to generate more interesting refinements.

Definition 2.8. Let $\uparrow$ and $\downarrow$ be two functions taking concept expressions as parameters and mapping them to finite sets of concepts. The *abstract refinement operator* is defined recursively on the structure of concept expressions, as follows.

$$\begin{aligned}
\zeta_{\uparrow,\downarrow}(A) &= \uparrow(A) \quad , A \in N_C \cup \{\top, \bot\} \ , \\
\zeta_{\uparrow,\downarrow}(\neg C) &= \uparrow(\neg C) \cup \{\neg C' \mid C' \in \zeta_{\downarrow,\uparrow}(C)\} \ , \\
\zeta_{\uparrow,\downarrow}(C \sqcap D) &= \uparrow(C \sqcap D) \cup \{C' \sqcap D \mid C' \in \zeta_{\uparrow,\downarrow}(C)\} \\
&\quad \cup \{C \sqcap D' \mid D' \in \zeta_{\uparrow,\downarrow}(D)\} \ , \\
\zeta_{\uparrow,\downarrow}(C \sqcup D) &= \uparrow(C \sqcup D) \cup \{C' \sqcup D \mid C' \in \zeta_{\uparrow,\downarrow}(C)\} \\
&\quad \cup \{C \sqcup D' \mid D' \in \zeta_{\uparrow,\downarrow}(D)\} \ , \\
\zeta_{\uparrow,\downarrow}(\forall R.C) &= \uparrow(\forall R.C) \cup \{\forall R.C' \mid C' \in \zeta_{\uparrow,\downarrow}(C)\} \ , \\
\zeta_{\uparrow,\downarrow}(\exists R.C) &= \uparrow(\exists R.C) \cup \{\exists R.C' \mid C' \in \zeta_{\uparrow,\downarrow}(C)\} \ .
\end{aligned}$$

Using the abstract refinement operator $\zeta_{\uparrow,\downarrow}$, we build two concrete refinement operators. The *generalization operator* and *specialization operator* are, respectively, defined as

$$\gamma_{\mathcal{O}} = \zeta_{\mathrm{UpCover}_{\mathcal{O}},\mathrm{DownCover}_{\mathcal{O}}} \ \text{ and } \ \rho_{\mathcal{O}} = \zeta_{\mathrm{DownCover}_{\mathcal{O}},\mathrm{UpCover}_{\mathcal{O}}} \ .$$

If we again revisit the example in Example 2.9, we can observe that $\gamma_{\mathcal{O}}(C \sqcup A) = \{\top, \top \sqcup A, C \sqcup A, C \sqcup B\}$ does contain the possible refinement $C \sqcup B$. Similarly, $\gamma_{\mathcal{O}}(\forall r.A) = \{\top, \forall r.A, \forall s.A, \forall r.B\}$ contains $\forall r.B$. Some basic properties of $\gamma_{\mathcal{O}}$ and $\rho_{\mathcal{O}}$ are shown in [64]. Most relevant of these is the fact that $\gamma_{\mathcal{O}}$ and $\rho_{\mathcal{O}}$ do in fact return, respectively, generalizations and specializations of the given concepts. Formally, for all concepts C and D, if $D \in \gamma_{\mathcal{O}}(C)$ then $C \sqsubseteq_{\mathcal{O}} D$. Similarly, if $D \in \rho_{\mathcal{O}}$ then $D \sqsubseteq_{\mathcal{O}} C$. Another important property to take note of is that both concrete refinement operators only return finite sets of refinements.

We define now the *axiom weakening operator* using these generalization and specialization operators.

Definition 2.9. Given an axiom α, the set of *weakenings* with respect to the ontology $\mathcal{O}$, written $g_{\mathcal{O}}(\alpha)$, is defined such that

$$\begin{aligned}
g_{\mathcal{O}}(C \sqsubseteq D) &= \{C' \sqsubseteq D \mid C' \in \rho_{\mathcal{O}}(C)\} \\
&\quad \cup \{C \sqsubseteq D' \mid D' \in \gamma_{\mathcal{O}}(D)\} \ , \\
g_{\mathcal{O}}(C(a)) &= \{C'(a) \mid C' \in \gamma_{\mathcal{O}}(C)\} \ .
\end{aligned}$$

The axioms in the set $g_{\mathcal{O}}(\alpha)$ are indeed weaker than α for every axiom α. This means that, given the reference ontology $\mathcal{O}$, α entails every axiom $\alpha' \in g_{\mathcal{O}}mc(\alpha)$, but the opposite is not necessarily true. That is, for each axiom $\alpha' \in g_{\mathcal{O}}(\alpha)$, $\alpha \models \mathcal{O}\alpha'$ holds.

Example 2.10. Let $N_C = \{A, B, C\}$, $N_I = \{a\}$ and $\mathcal{O} = \{A \sqsubseteq B, A \sqsubseteq C\}$. $\mathrm{sub}(\mathcal{O}) = \{\top, \bot, A, B, C\}$. $\gamma_{\mathcal{O}}(A) = \{A, B, C\}$ and $\rho_{\mathcal{O}}(B) = \{B, A\}$. The weakenings of $B \sqsubseteq A$ with respect to $\mathcal{O}$ are therefore $g_{\mathcal{O}}(B \sqsubseteq A) = \{B \sqsubseteq A, B \sqsubseteq B, B \sqsubseteq C, A \sqsubseteq A\}$. Similarly, $g_{\mathcal{O}}(A(a)) = \{A(a), B(a), C(a)\}$.

In order to apply the weakening operator to the problem of repairing inconsistent ontologies, we follow the algorithm depicted in Algorithm 2.1. First, as mentioned above, the computation of a non-trivial upward and downward cover set is only possible when using a consistent ontology. We, therefore, need to first find a consistent

subontology $\mathcal{O}^{\text{ref}}$ of $\mathcal{O}$ to serve as *reference ontology*. One possibility outlined in [64] is to select a random maximal consistent subset of $\mathcal{O}$. Another option is to choose the intersection of some (or all) maximal consistent subsets of $\mathcal{O}$ (e.g., [44]). This latter option has the advantage of using only information for the weakening that is likely to be correct. On the other hand, it will lead to diminished upward and downward covers and therefore can reduce the number of possible weakenings.

Once a reference ontology has been chosen, and as long as $\mathcal{O}$ is inconsistent, we continue by selecting a "bad axiom" α, remove it from $\mathcal{O}$, and replace it with a weaker axiom from $g_{\mathcal{O}^{\text{ref}}}(\alpha)$. As for the choice of reference ontology, there are also multiple options for the selection of bad axioms. [64] proposes two different implementations. One is based on choosing axioms at random, and another implementation randomly samples some (or all) minimal inconsistent subsets of axioms $J_1, J_2, \ldots, J_k \subseteq \mathcal{O}$ and returns one axiom from the ones occurring the most often. The weaker axiom is chosen from the set of weakening by selecting one uniformly at random. While there may be better heuristics for choosing the weakening, termination of the algorithm is not generally guaranteed. It has, however, been shown in [16], that when selecting among the weakening at random using a uniform distribution, their algorithm is almost surely going to terminate. It has yet to be shown whether this holds also for the variation presented in this paper. A similar effect of almost sure termination could also be achieved by maintaining a constant non-zero probability of removing the axiom at each step.

Algorithm 2.1 RepairOntologyWeaken($\mathcal{O}$)

$\mathcal{O}^{\text{ref}} \leftarrow$ FindConsistentSubset($\mathcal{O}$)
while $\mathcal{O}$ is inconsistent **do**
  $\alpha_{\text{bad}} \leftarrow$ FindBadAxiom($\mathcal{O}$)
  $\alpha_{\text{weaker}} \leftarrow$ SelectWeakerAxiom($g_{\mathcal{O}^{\text{ref}}}(\alpha_{\text{bad}})$)
  $\mathcal{O} \leftarrow (\mathcal{O} \setminus \{\alpha_{\text{bad}}\}) \cup \{\alpha_{\text{weaker}}\}$
end while
Return $\mathcal{O}$

Example 2.11. Given the inconsistent ontology $\mathcal{O} = \{A \sqsubseteq B \sqcap \neg C, B \sqsubseteq C, A(a)\}$, we want to repair it such that it becomes consistent. We select first a maximal consistent subset as reference ontology $\mathcal{O}^{\text{ref}} = \{B \sqsubseteq C, A(a)\}$. To find a bad axiom, we sample all minimal inconsistent subsets. Since removing any of the axioms makes it consistent, there is only one minimal inconsistent subset $J_1 = \mathcal{O}$. Say we choose the axiom $A \sqsubseteq B \sqcap \neg C$ and weaken it to $A \sqsubseteq C \sqcap \neg C$. The ontology is still inconsistent, with the single minimal inconsistent subset $J_1 = \{A \sqsubseteq C \sqcap \neg C, A(a)\}$. Let us choose to weaken $A(a)$ to $\top(a)$. The ontology is now consistent.

While the repair using axiom weakening is capable of preserving more information than removal, as has been empirically shown in [64], this is influenced significantly by the choice of which axioms to weaken and which weaker axioms to use. In fact, computing a maximal consistent subset may in some cases lead to a more gentle repair than applying the procedure outlined above in Algorithm 2.1. This can be seen in Example 2.11, where the last replacement alone would have restored consistency, and the information lost in the previous replacement could have been preserved.

Chapter 3

Extending Axiom Weakening

It follows an explanation and definition of how this idea of axiom weakening can also be extended to more expressive DLs. Concretely, an extension of the refinement operators and axiom weakening operator will be proposed that can be used with $\mathcal{SROIQ}$ ontologies.

3.1 Difficulties with Weakening $\mathcal{SROIQ}$

The main difficulties that arise when weakening axioms in $\mathcal{SROIQ}$ ontologies, and especially when weakening RIAs, are related to ensuring that the constraints on the use of non-simple roles and the regularity of the RBox as a whole are maintained. In many logics, the union of two ontologies would be guaranteed to be another valid ontology. While this is the case for FOL theories or even $\mathcal{ALC}$ ontologies, it is not true for $\mathcal{SROIQ}$ ontologies. As mentioned in Section 2.3.2, merging two $\mathcal{SROIQ}$ ontologies may create an ontology that violates these constraints. While these restrictions help ensure the decidability of the logic, they also make $\mathcal{SROIQ}$ more difficult to use. Not only must these complications be taken into account when authoring ontologies, but they must be carefully considered also when modifying them using automatic transformations, such as applying axiom weakening based repair algorithms. Replacing an axiom with another axiom, even if the new axiom is weaker, can cause the resulting ontology to violate one or more of the global constraints in $\mathcal{SROIQ}$. This may happen even if the new axiom on its own is valid, as can be seen in Example 3.1 and Example 3.2.

Example 3.1. Take the ontology $\mathcal{O} = \{r \circ s \circ r \sqsubseteq t, r \sqsubseteq s, \top \sqsubseteq \forall t.\bot, \exists s.Self \sqsubseteq \top\}$. The extension of t is empty in every model of this ontology. It follows that, ignoring the global constraints, the axiom $r \sqsubseteq s$ could be weakened to $t \sqsubseteq s$. This would, however, result in an ontology where s is non-simple, which is not allowed since s is used as part of a $Self$ constraint. Further, the RBox of the resulting ontology will not be regular, since for any preorder $\preceq$, $t \preceq s$ must hold for the new axiom, but $t \not\preceq s$ is required by the complex RIA.

Example 3.2. As another example, with less trivial weakening, take the ontology $\mathcal{O} = \{r \circ s \circ r \sqsubseteq t, \top \sqsubseteq \forall t.\{a\}, \top \sqsubseteq \exists s.\{a\}\}$. Since the range of t is restricted to the single individual a, and s contains all connections to a, $t \sqsubseteq_{\mathcal{O}} s$. The axiom $r \circ s \circ r \sqsubseteq t$ could therefore be weakened to $r \circ s \circ r \sqsubseteq s$. Yet, this would result in a non-regular RBox.

The weakening proposed in [64] and described above in Section 2.5 operates on $\mathcal{ALC}$ ontologies. A similar approach can also be used for more expressive logics like $\mathcal{ALCHOIQ}$, without additional constraints. In order to extend axiom weakening to $\mathcal{SROIQ}$ ontologies, however, it is necessary to restrict the allowed weakening accordingly, to ensure that these issues are avoided. In [16] an axiom weakening operator is proposed for parts of $\mathcal{SROIQ}$. However, to prevent the aforementioned issues, some restrictions are put in place. For example, the refinement of RIAs has not been considered at all. For this thesis, however, the axiom weakening operator has been extended to handle also RIAs. To achieve this, it must be ensured that when weakening disjoint role axioms or refining cardinality and $Self$ constraints, only simple roles are used. Further, it must be guaranteed that all roles that are currently used in such contexts remain simple even after adding the weakened axioms to the ontology. Finally, the regularity of the RBox must be maintained by the addition of the weakened axioms. We now discuss which restrictions we applied to satisfy these requirements.

Firstly, the covers and refinement operators for roles are defined to operate only on simple roles. A similar restriction has already been applied to the refinement operators suggested in [16]. Restricting the refinement to simple roles guarantees that the new axioms created by weakening will not contain non-simple roles in axioms or concepts where simple roles are required. An important detail that has not been considered in [16] is that the roles over which the covers operate must be simple in all ontologies that the weaker axioms will be used in. It is therefore generally not sufficient to use all the roles that are simple in the reference ontology, since the reference ontology may not contain all RBox axioms, and therefore contain simple roles that are not simple in the full ontology.

Example 3.3. Given the inconsistent ontology $\mathcal{O} = \{s \circ s \sqsubseteq s, s \sqsubseteq r, t \sqsubseteq r, \top \sqsubseteq \forall r.\bot, \top \sqsubseteq \exists s.\top, \top \sqsubseteq \exists t.Self\}$, we may choose as reference ontology the maximal consistent subset $\mathcal{O}^{\text{ref}} = \mathcal{O} \setminus \{s \sqsubseteq r\}$. In this case, r is simple in $\mathcal{O}^{\text{ref}}$, however, it is not allowed to weaken $\top \sqsubseteq \exists t.Self$ to $\top \sqsubseteq \exists r.Self$, because r is not simple in $\mathcal{O} \setminus \{\top \sqsubseteq \exists t.Self\} \cup \{\top \sqsubseteq \exists r.Self\}$ and can therefore not be used in $Self$ constraints.

For this reason, we give to the upward and downward cover as an argument not only the reference ontology $\mathcal{O}^{\text{ref}}$, but also the full ontology $\mathcal{O}^{\text{full}}$. The same must be done for the axiom weakening operator that uses the cover functions. Both $\mathcal{O}^{\text{ref}}$ and $\mathcal{O}^{\text{full}}$ share the same vocabulary N_I, N_C, and N_R. We assume that all roles that are simple in $\mathcal{O}^{\text{full}}$ are also simple in $\mathcal{O}^{\text{ref}}$. While this is not strictly required, it will be true for all algorithms shown in this thesis that $\mathcal{O}^{\text{ref}} \subseteq \mathcal{O}^{\text{full}}$. In the context of repairing ontologies, $\mathcal{O}^{\text{full}}$ can be chosen to be the unmodified ontology that we want to repair.

An alternative to always choosing simple roles is to define two versions of the covers, one which operators only over simple roles and one which uses also non-simple roles. Then in a context in which simple roles are required, the more restrictive cover is used, while in all other cases, the complete covers can be employed. While this variation has been implemented, it has not been evaluated and will not be further considered in the rest of the thesis.

Another assumption made in [16] is that the existential and universal roles are simple, and may therefore be included in the upward and downward covers. While this is not a significant restriction in theory, it contradicts the OWL 2 DL specification, which directly defines these roles to be non-simple in every ontology. Including the existential and universal roles in the covers, while augmenting the number of possible weakening, is not necessarily required. Therefore, this thesis will consider, as stated in Section 2.1.1, the universal role to be non-simple. This allows the implementation of

the proposed approach using existing highly-optimized reasoners designed for OWL 2 DL. It has to be considered, however, that removing the existential and universal role from the covers can prevent the weakening of RIAs, disjoint role axioms and role assertions to tautologies. Removal of these axioms may be required if they are by themselves inconsistent, meaning that not being able to weaken them to tautologies could mean the repair algorithm never terminates. For this reason, the weakening operator for these axioms explicitly includes a tautological axiom. Note that for the case of positive and negative role assertions, the extension of the covers to include also non-simple roles would be an alternative solution. This is however not the case for disjoint role assertions, as the roles used in them must always be simple.

3.2 Weakening Role Inclusion Axioms

Let us now consider the weakening of simple and complex RIAs. To ensure that by adding weakened axioms we do not cause a constraint violation in existing axioms and concepts, we choose the allowed weakening for RIAs such that all roles that are simple in $\mathcal{O}^{\text{full}}$, are also simple after adding to it a weakening of one of its axioms. We observe that for complex RIAs $S_1 \circ \cdots \circ S_n \sqsubseteq R$ we should not refine the role R. Since all roles returned by our refinement operator will be simple in $\mathcal{O}^{\text{full}}$, replacing R with a refinement R' would create a new axiom which, when added to the ontology, would make the role R' which was simple in $\mathcal{O}^{\text{full}}$ non-simple. A similar argument can be made for refining R in a simple RIA $S \sqsubseteq R$ where the role S is non-simple in $\mathcal{O}^{\text{full}}$. Therefore, The only case in which it is possible to refine the super role of a RIA during axiom weakening requires that the RIA is simple and additionally that the sub role of the RIA is simple in $\mathcal{O}^{\text{full}}$.

When it comes to refining the left-hand side of RIAs, we do not need any special restrictions. The important observation here is that all roles that are returned by the refinement operators will be simple. This means that in a simple RIA $R \sqsubseteq S$, even if S is simple, replacing R with another simple role will not cause S to become non-simple. For a complex RIA $S_1 \circ \cdots \circ S_n \sqsubseteq R$ on the other hand, the role R must already have been non-simple in $\mathcal{O}^{\text{full}}$, and replacing any S_i with a refinement does not affect whether a role is simple or non-simple.

A more interesting question is whether such a weakening may still cause a non-regular RBox. The main insight is that simple roles are always allowed on the left-hand side of a RIA. While this is more directly evident in some alternative definitions of regularity (e.g., [56]) it is not so apparent from the one presented in this thesis. Intuitively, the constraints given in Section 2.1.1 for regularity disallow dependency cycles that contain complex RIAs. Simple roles cannot be part of such a cycle, since the cycle must contain at least one complex RIA to be a violation of the constraint, and all roles that depend in this sense on a complex RIA must be non-simple. A more formal justification for this fact is given in the proof for Lemma 3.4. Since all refinements of the left-hand side of RIAs are performed using simple roles, these cannot lead to a non-regular RBox. Further, refinements of the super role of RIAs are only performed on simple RIAs $S \sqsubseteq R$ where S is a simple role. Since S is simple in this case, all refinements of R are allowed, potentially also if the refinement yielded a non-simple role.

Note that when using this approach for RIAs in an ontology in which all roles are simple, such as in $\mathcal{ALCH}$, we get the obvious weakening operator that specializes the left-hand side and generalizes the right-hand side, without any further restrictions.

3.2.1 An Alternative Approach

Instead of limiting the weakening of RIAs in this way, an alternative approach that may be considered is to generate axioms and then filter out those that would violate some restrictions. The axioms can be generated by using the specialization operator on all roles on the left-hand side and the generalization operator on the role on the right-hand side. Say we still want to keep all simple roles of $\mathcal{O}^{\text{full}}$ simple after the addition of the weakened RIA. Then we would filter out all complex RIA in which the right-hand side became a role that is supposed to be simple. Further, all simple RIA in which the left-hand side is not guaranteed to be simple, but the right-hand side must be, should be removed.

Additionally, to guarantee the regularity of the RBox, a preorder $\preceq$ may be fixed before the start of the repair. The preorder must be a witness for the regularity of the ontology $\mathcal{O}^{\text{full}}$. Then, when weakening RIAs, they must be filtered to conform to the preorder $\preceq$. This approach will ensure regularity since all RIAs that may be added will conform to the chosen preorder. All existing RIAs trivially conform to it, given it is a precondition for the choice of $\preceq$. One important detail is that the preorder may not be changed in between a repair. It is not generally safe to combine a weakened RIA generated using one preorder with another RIA that used another preorder during weakening. This mechanism has been implemented, but not evaluated or further explored in this thesis.

3.3 Axiom Weakening in $\mathcal{SROIQ}$

It follows now the extended definitions for the cover sets, refinement operators, and axiom weakening operator covering $\mathcal{SROIQ}$ concepts and axioms. This work builds on the approaches presented in [16] and introduces further the weakening of RIAs as well as a separation between $\mathcal{O}^{\text{ref}}$ and $\mathcal{O}^{\text{full}}$. We begin by defining the upward and downward cover sets. These are now defined for both concepts and roles. The set of subconcepts is taken from $\mathcal{O}^{\text{full}}$ as opposed to $\mathcal{O}^{\text{ref}}$, as it is likely to contain more subconcepts, thus improving the granularity of the cover sets. Further, the cover sets for roles are computed over the set of simple roles in $\mathcal{O}^{\text{full}}$. Additionally, we define the upward and downward covers also for non-negative integers, as they will be useful for the refinement of cardinality constraints.

Definition 3.1. Let $\mathcal{O}^{\text{full}}$ and $\mathcal{O}^{\text{ref}} \subseteq \mathcal{O}^{\text{full}}$ be two $\mathcal{SROIQ}$ ontologies that share the same vocabulary N_C, N_R, and N_I. The *upward cover* and *downward cover* for a concept C are given by

$$\begin{aligned}\text{UpCover}_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(C) = \{&D \in \text{sub}(\mathcal{O}^{\text{full}}) \mid C \sqsubseteq_{\mathcal{O}^{\text{ref}}} D \text{ and} \\ &\nexists D' \in \text{sub}(\mathcal{O}^{\text{full}}) \text{ with } C \sqsubset_{\mathcal{O}^{\text{ref}}} D' \sqsubset_{\mathcal{O}^{\text{ref}}} D\} \ , \\ \text{DownCover}_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(C) = \{&D \in \text{sub}(\mathcal{O}^{\text{full}}) \mid D \sqsubseteq_{\mathcal{O}^{\text{ref}}} C \text{ and} \\ &\nexists D' \in \text{sub}(\mathcal{O}^{\text{full}}) \text{ with } D \sqsubset_{\mathcal{O}^{\text{ref}}} D' \sqsubset_{\mathcal{O}^{\text{ref}}} C\} \ .\end{aligned}$$

The upward and downward covers for a role R are given by

$$\begin{aligned}\text{UpCover}_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(R) = \{&S \in \mathcal{L}(N_R) \mid R \sqsubseteq_{\mathcal{O}^{\text{ref}}} S \text{ and} \\ &\nexists S' \in \mathcal{L}(N_R) \text{ with } R \sqsubset_{\mathcal{O}^{\text{ref}}} S' \sqsubset_{\mathcal{O}^{\text{ref}}} S \text{ and} \\ &S, S' \text{ are simple in } \mathcal{O}^{\text{full}}\} \ ,\end{aligned}$$

$$\begin{aligned}\mathrm{DownCover}_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(R) = \{S \in \mathcal{L}(N_R) \mid\ & S \sqsubseteq_{\mathcal{O}^{\text{ref}}} R \text{ and} \\ & \nexists S' \in \mathcal{L}(N_R) \text{ with } S \sqsubset_{\mathcal{O}^{\text{ref}}} S' \sqsubset_{\mathcal{O}^{\text{ref}}} R \text{ and} \\ & S, S' \text{ are simple in } \mathcal{O}^{\text{full}}\}\ .\end{aligned}$$

The upward and downward covers for a non-negative integer n are given by

$$\begin{aligned}\mathrm{UpCover}_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(n) &= \{n, n+1\}\ , \\ \mathrm{DownCover}_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(n) &= \begin{cases}\{n\} & \text{if } n = 0 \\ \{n, n-1\} & \text{if } n > 0\end{cases}\ .\end{aligned}$$

As has already been observed in Example 2.9, on their own, the upward and downward covers of concepts are missing some interesting refinements. Since they operate only over the subconcepts of $\mathcal{O}^{\text{full}}$, some rather obvious refinements, like for example removing conjuncts, will often be missed.

Example 3.4. Let $N_C = \{A, B, C\}$, $N_R = \{r, s\}$, and $\mathcal{O} = \{A \sqsubseteq B, r \sqsubseteq s\}$. $\mathrm{sub}(\mathcal{O}) = \{\top, \bot, A, B\}$. The upward cover of $C \sqcup A$ is equal to $\mathrm{UpCover}_{\mathcal{O},\mathcal{O}}(C \sqcup A) = \{\top\}$. The potential refinement to $C \sqcup B$ will be missed by all upward and downward covers because $C \sqcup B \notin \mathrm{sub}(\mathcal{O})$. Further, $\mathrm{UpCover}_{\mathcal{O},\mathcal{O}}(\forall r.A) = \{\top\}$, even if $\forall r.B$ and $\forall s.A$ are both possible generalizations.

To also address some of these omissions, we define generalization and specialization operators that exploit the recursive structure of the concept being refined to generate more complex refinements. For convenience, we also define these operators for roles, even though their refinement operators are fully equivalent to the upward and downward cover functions.

Definition 3.2. Let $\uparrow$ and $\downarrow$ be two functions with domain $\mathcal{L}(N_C, N_R, N_I) \cup \mathcal{L}(N_R) \cup \mathbb{N}_0$. They map every concept to a finite subset of $\mathcal{L}(N_C, N_R, N_I)$, every role to a subset of $\mathcal{L}(N_R)$, and every non-negative integer to a finite subset of $\mathbb{N}_0$. The *abstract refinement operator* is defined recursively on the structure of concepts, as follows.

$$\begin{aligned}\zeta_{\uparrow,\downarrow}(A) &= \uparrow(A) \quad , A \in N_C \cup \{\top, \bot\}\ , \\ \zeta_{\uparrow,\downarrow}(\neg C) &= \uparrow(\neg C) \cup \{\neg C' \mid C' \in \zeta_{\downarrow,\uparrow}(C)\}\ , \\ \zeta_{\uparrow,\downarrow}(C \sqcap D) &= \uparrow(C \sqcap D) \cup \{C' \sqcap D \mid C' \in \zeta_{\uparrow,\downarrow}(C)\} \cup \{C \sqcap D' \mid D' \in \zeta_{\uparrow,\downarrow}(D)\}\ , \\ \zeta_{\uparrow,\downarrow}(C \sqcup D) &= \uparrow(C \sqcup D) \cup \{C' \sqcup D \mid C' \in \zeta_{\uparrow,\downarrow}(C)\} \cup \{C \sqcup D' \mid D' \in \zeta_{\uparrow,\downarrow}(D)\}\ , \\ \zeta_{\uparrow,\downarrow}(\forall R.C) &= \uparrow(\forall R.C) \cup \{\forall R'.C \mid R' \in \zeta_{\downarrow,\uparrow}(R)\} \cup \{\forall R.C' \mid C' \in \zeta_{\uparrow,\downarrow}(C)\}\ , \\ \zeta_{\uparrow,\downarrow}(\exists R.C) &= \uparrow(\exists R.C) \cup \{\exists R'.C \mid R' \in \zeta_{\uparrow,\downarrow}(R)\} \cup \{\exists R.C' \mid C' \in \zeta_{\uparrow,\downarrow}(C)\}\ , \\ & \mathcal{SROIQ} \text{ concepts:} \\ \zeta_{\uparrow,\downarrow}(\{a\}) &= \uparrow(\{a\})\ , \\ \zeta_{\uparrow,\downarrow}(\exists R.\mathit{Self}) &= \uparrow(\exists R.\mathit{Self}) \cup \{\exists R'.\mathit{Self} \mid R' \in \zeta_{\uparrow,\downarrow}(R)\}\ , \\ \zeta_{\uparrow,\downarrow}(\geq n\ R.C) &= \uparrow(\geq n\ R.C) \cup \{\geq n\ R'.C \mid R' \in \zeta_{\uparrow,\downarrow}(R)\} \\ & \quad \cup \{\geq n\ R.C' \mid C' \in \zeta_{\uparrow,\downarrow}(C)\} \cup \{\geq n\ 'R.C \mid n' \in \downarrow(n)\}\ , \\ \zeta_{\uparrow,\downarrow}(\leq n\ R.C) &= \uparrow(\leq n\ R.C) \cup \{\leq n\ R'.C \mid R' \in \zeta_{\downarrow,\uparrow}(R)\} \\ & \quad \cup \{\leq n\ R.C' \mid C' \in \zeta_{\downarrow,\uparrow}(C)\} \cup \{\leq n\ 'R.C \mid n' \in \uparrow(n)\}\ , \\ & \mathcal{SROIQ} \text{ roles:} \\ \zeta_{\uparrow,\downarrow}(R) &= \uparrow(R)\ .\end{aligned}$$

From the abstract refinement operator $\zeta_{\uparrow,\downarrow}$, two concrete refinement operators, the *generalization operator* and *specialization operator* are, respectively, defined as

$$\gamma_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}} = \zeta_{\text{UpCover}_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}},\text{DownCover}_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}} \quad \text{and}$$
$$\rho_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}} = \zeta_{\text{DownCover}_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}},\text{UpCover}_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}} \quad .$$

Revisiting the case in Example 3.4 we observe that $\gamma_{\mathcal{O},\mathcal{O}}(C \sqcup A) = \{\top, \top \sqcup A, C \sqcup A, C \sqcup B\}$ does contain $C \sqcup B$ as a possible refinement. Further, $\gamma_{\mathcal{O},\mathcal{O}}(\forall r.A) = \{\top, \forall r.A, \forall s.A, \forall r.B\}$ contains both $\forall r.B$ and $\forall s.A$. We will now show some basic properties of $\gamma_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}$ and $\rho_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}$ that will prove useful in the remainder of this thesis. For one, the sets of generalizations or specializations produced by the concrete refinement operators are finite. This is important for the implementation of the operator to be able to compute all refinements. Further, we show that the concrete refinement operators are in fact generalization and specialization operators.

Lemma 3.1. *For every pair of $\mathcal{SROIQ}$ ontologies $\mathcal{O}^{\text{ref}}, \mathcal{O}^{\text{full}}$ and every pair of concepts or roles $X, Y \in \mathcal{L}(N_C, N_R, N_I) \cup \mathcal{L}(N_R)$:*

1. ***generalization finiteness:*** *$\gamma_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(X)$ is finite*
   specialization finiteness: *$\rho_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(X)$ is finite*
2. ***generalization:*** *if $X \in \gamma_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(Y)$ then $Y \sqsubseteq_{\mathcal{O}^{\text{ref}}} X$*
   specialization: *if $X \in \rho_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(Y)$ then $X \sqsubseteq_{\mathcal{O}^{\text{ref}}} Y$*

Proof.

1. **finiteness:** The upward and downward cover sets for concepts, roles, and integers are always finite. The number of roles is finite because the set of role names is finite. Since refinement of roles is equivalent to applying the upward or downward covers, the set of refinements must equally be finite. Similarly, the covers over concepts are limited to subconcepts, and since ontologies are finite sets of axioms, and all axioms have a finite size, there must be a finite number of subconcepts. We can show, by structural induction on concept expressions, that the result of the refinement operators must be a finite set for all concepts. This follows directly from the fact that the result is always the union of a constant number of finite sets.

2. **generalization and specialization:** We first observe that by definition, if $X \in \text{UpCover}_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(Y)$ then $Y \sqsubseteq_{\mathcal{O}^{\text{ref}}} X$ must hold. Likewise, it follows from the definition that if $X \in \text{DownCover}_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(Y)$ then $X \sqsubseteq_{\mathcal{O}^{\text{ref}}} Y$. For an integer n, the upward cover contains only values greater or equal to n, while the downward cover contains only values less or equal to n. We proceed with a proof by structural induction. We will show the proof for the generalization operator, the proof for the specialization operator can be approached analogously.

   - For $Y \in N_C \cup \mathcal{L}(N_R) \cup \{\{a\} \mid a \in N_I\} \cup \{\top, \bot\}$ the generalization operator is equivalent to a simple application of the upwards cover and $Y \sqsubseteq_{\mathcal{O}^{\text{ref}}} X$ holds.
   - For $Y = \neg C$, given any model I of $\mathcal{O}^{\text{ref}}$, we know for all elements $\delta \in (\neg C)^I$ that $\delta \notin C^I$. By our inductive hypothesis, we know that $C' \sqsubseteq_{\mathcal{O}^{\text{ref}}} C$ holds for $C' \in \rho_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(C)$. It follows that $\delta \notin C'^I$ and therefore $\delta \in (\neg C')^I$. Therefore, $(\neg C)^I \subseteq (\neg C')^I$ in every model I of $\mathcal{O}^{\text{ref}}$ and consequently $\neg C \sqsubseteq_{\mathcal{O}^{\text{ref}}} \neg C'$.
   - For $Y = C \sqcap D$, given any model I of $\mathcal{O}^{\text{ref}}$, we know for all elements $\delta \in (C \sqcap D)^I$ that $\delta \in C^I$ and $\delta \in D^I$. By our inductive hypothesis, we know that

$C \sqsubseteq_{\mathcal{O}^{\text{ref}}} C'$ holds for $C' \in \gamma_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(C)$ and $D \sqsubseteq_{\mathcal{O}^{\text{ref}}} D'$ for $D' \in \gamma_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(D)$. It follows that $\delta \in C'^I$ and $\delta \in D'^I$, therefore $\delta \in (C' \sqcap D)^I$ and $\delta \in (C \sqcap D')^I$. We conclude that $(C \sqcap D)^I \subseteq (C' \sqcap D)^I$ and $(C \sqcap D)^I \subseteq (C \sqcap D')^I$ in every model I of $\mathcal{O}^{\text{ref}}$ and consequently $C \sqcap D \sqsubseteq_{\mathcal{O}^{\text{ref}}} C' \sqcap D$ and $C \sqcap D \sqsubseteq_{\mathcal{O}^{\text{ref}}} C \sqcap D'$.

- For $Y = C \sqcup D$, given any model I of $\mathcal{O}^{\text{ref}}$, we know for all elements $\delta \in (C \sqcup D)^I$ that $\delta \in C^I \cup D^I$. By our inductive hypothesis, we know that $C \sqsubseteq_{\mathcal{O}^{\text{ref}}} C'$ holds for $C' \in \gamma_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(C)$ and $D \sqsubseteq_{\mathcal{O}^{\text{ref}}} D'$ for $D' \in \gamma_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(D)$. It follows that $\delta \in C'^I \cup D^I$ and $\delta \in C^I \cup D'^I$, therefore $\delta \in (C' \sqcup D)^I$ and $\delta \in (C \sqcup D')^I$. We conclude that $(C \sqcup D)^I \subseteq (C' \sqcup D)^I$ and $(C \sqcup D)^I \subseteq (C \sqcup D')^I$ in every model I of $\mathcal{O}^{\text{ref}}$ and consequently, $C \sqcup D \sqsubseteq_{\mathcal{O}^{\text{ref}}} C' \sqcup D$ and $C \sqcup D \sqsubseteq_{\mathcal{O}^{\text{ref}}} C \sqcup D'$.

- For $Y = \forall R.C$, given any model I of $\mathcal{O}^{\text{ref}}$, we know for all elements $\delta \in (\forall R.C)^I$ that for all δ' such that $\langle \delta, \delta' \rangle \in R^I$, $\delta' \in C^I$. By our inductive hypothesis, we know that $C \sqsubseteq_{\mathcal{O}^{\text{ref}}} C'$ holds for $C' \in \gamma_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(C)$ and $R' \sqsubseteq_{\mathcal{O}^{\text{ref}}} R$ for $R' \in \rho_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(R)$. It follows that $R'^I \subseteq R^I$, so there are no new candidates for δ'. We further know that for all elements $\delta' \in C^I$, $\delta' \in C'^I$. We conclude that $\delta \in (\forall R'.C)^I$ and $\delta \in (\forall R.C')^I$. Therefore, $\forall R.C \sqsubseteq_{\mathcal{O}^{\text{ref}}} \forall R'.C$ and $\forall R.C \sqsubseteq_{\mathcal{O}^{\text{ref}}} \forall R.C'$.

- For $Y = \exists R.C$, given any model I of $\mathcal{O}^{\text{ref}}$, we know for all elements $\delta \in (\exists R.C)^I$ there exists a δ' such that $\langle \delta, \delta' \rangle \in R^I$ and $\delta' \in C^I$. By our inductive hypothesis, we know that $C \sqsubseteq_{\mathcal{O}^{\text{ref}}} C'$ holds for $C' \in \gamma_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(C)$ and $R \sqsubseteq_{\mathcal{O}^{\text{ref}}} R'$ for $R' \in \gamma_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(R)$. It follows that $\langle \delta, \delta' \rangle \in R'^I$ and $\delta' \in C'^I$. We conclude that $\delta \in (\exists R'.C)^I$ and $\delta \in (\exists R.C')^I$. Hence, $\exists R.C \sqsubseteq_{\mathcal{O}^{\text{ref}}} \exists R'.C$ and $\exists R.C \sqsubseteq_{\mathcal{O}^{\text{ref}}} \exists R.C'$.

- For $Y = \exists R.\mathit{Self}$, given any model I of $\mathcal{O}^{\text{ref}}$, we know for all elements $\delta \in (\exists R.\mathit{Self})^I$, $\langle \delta, a \rangle \in R^I$. By our inductive hypothesis, we know that $R \sqsubseteq_{\mathcal{O}^{\text{ref}}} R'$ for $R' \in \gamma_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(R)$. It follows that $\langle \delta, \delta \rangle \in R'^I$. We conclude that $\delta \in (\exists R'.\mathit{Self})^I$ and thus, $\exists R.\mathit{Self} \sqsubseteq_{\mathcal{O}^{\text{ref}}} \exists R'.C\mathit{Self}$.

- For $Y = \geq n\ R.C$, given any model I of $\mathcal{O}^{\text{ref}}$, we know for all elements $\delta \in (\geq n\ R.C)^I$ there exist at least n elements $\delta_1, \ldots, \delta_n$ such that $\langle \delta, \delta_i \rangle \in R^I$ and $\delta_i \in C^I$ for $i = 1, \ldots, n$. Since there are at least n such elements, there are also at least $k \leq n$, meaning $\geq n\ R.C \sqsubseteq_{\mathcal{O}^{\text{ref}}} \geq k\ R.C$. By our inductive hypothesis, we also know that $C \sqsubseteq_{\mathcal{O}^{\text{ref}}} C'$ holds for $C' \in \gamma_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(C)$ and $R \sqsubseteq_{\mathcal{O}^{\text{ref}}} R'$ for $R' \in \gamma_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(R)$. It follows that $\langle \delta, \delta_i \rangle \in R'^I$ and $\delta_i \in C'^I$ for $i = 1, \ldots, n$. We conclude that $\delta \in (\geq n\ R'.C)^I$ and $\delta \in (\geq n\ R.C')^I$. Hence, $\geq n\ R.C \sqsubseteq_{\mathcal{O}^{\text{ref}}} \geq n\ R'.C$ and $\geq n\ R.C \sqsubseteq_{\mathcal{O}^{\text{ref}}} \geq n\ R.C'$.

- For $Y = \leq n\ R.C$, given any model I of $\mathcal{O}^{\text{ref}}$, we know for all elements $\delta \in (\leq n\ R.C)^I$ there exist at most n elements $\delta_1, \ldots, \delta_n$ such that $\langle \delta, \delta_i \rangle \in R^I$ and $\delta_i \in C^I$ for $i = 1, \ldots, n$. Since there are at most n such elements, there are also at most $k \geq n$, meaning $\leq n\ R.C \sqsubseteq_{\mathcal{O}^{\text{ref}}} \leq k\ R.C$. By our inductive hypothesis, we also know that $C' \sqsubseteq_{\mathcal{O}^{\text{ref}}} C$ holds for $C' \in \rho_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(C)$ and $R' \sqsubseteq_{\mathcal{O}^{\text{ref}}} R$ for $R' \in \rho_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(R)$. It follows that there exist no other $\delta' \notin \{\delta_1, \ldots, \delta_n\}$ such that $\langle \delta, \delta' \rangle \in R'^I$ and $\delta' \in C$, or $\langle \delta, \delta' \rangle \in R^I$ and $\delta' \in C'^I$. We conclude that there are still at most n such elements, hence, $\delta \in (\leq n\ R'.C)^I$ and $\delta \in (\leq n\ R.C')^I$. Therefore, $\leq n\ R.C \sqsubseteq_{\mathcal{O}^{\text{ref}}} \leq n\ R'.C$ and $\leq n\ R.C \sqsubseteq_{\mathcal{O}^{\text{ref}}} \leq n\ R.C'$.

□

Using the specialization and generalization operators, we can now define the *axiom weakening operator.*

Definition 3.3. Given an axiom α, the set of *weakenings* with respect to the reference ontology $\mathcal{O}^{\text{ref}}$ and full ontology $\mathcal{O}^{\text{full}}$, written $g_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(\alpha)$ is defined such that

$$
\begin{aligned}
g_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(C \sqsubseteq D) &= \{C' \sqsubseteq D \mid C' \in \rho_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(C)\} \\
&\quad \cup \{C \sqsubseteq D' \mid D' \in \gamma_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(D)\} \ , \\
g_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(C(a)) &= \{C'(a) \mid C' \in \gamma_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(C)\} \ , \\
&\quad \mathcal{SROIQ} \text{ axioms:} \\
g_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(R(a,b)) &= \{R'(a,b) \mid R' \in \gamma_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(R)\} \cup \{R(a,b), \bot \sqsubseteq \top\} \ , \\
g_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(\neg R(a,b)) &= \{\neg R'(a,b) \mid R' \in \rho_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(R)\} \cup \{\neg R(a,b), \bot \sqsubseteq \top\} \ , \\
g_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(a = b) &= \{a = b, \bot \sqsubseteq \top\} \ , \\
g_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(a \neq b) &= \{a \neq b, \bot \sqsubseteq \top\} \ , \\
g_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(\text{disjoint}(R,S)) &= \{\text{disjoint}(R', S) \mid R' \in \rho_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(R)\} \\
&\quad \cup \{\text{disjoint}(R, S') \mid S' \in \rho_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(S)\} \\
&\quad \cup \{\text{disjoint}(R,S), \bot \sqsubseteq \top\} \ , \\
g_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(S_1 \circ \cdots \circ S_n \sqsubseteq R) &= \{S_1 \circ \cdots \circ S_i' \circ \cdots \circ S_n \sqsubseteq R \\
&\quad \mid S_i' \in \rho_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(S_i) \text{ for } i = 1, \ldots, n\} \\
&\quad \cup \{S_1 \sqsubseteq R' \mid R' \in \gamma_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}} \text{ and } n = 1 \text{ and} \\
&\quad\quad S_1 \text{ is simple in } \mathcal{O}^{\text{full}}\} \\
&\quad \cup \{S_1 \circ \cdots \circ S_n \sqsubseteq R, \bot \sqsubseteq \top\} \ .
\end{aligned}
$$

As was the case for the axiom weakening operator in $\mathcal{ALC}$, the axioms α' returned by $g_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(\alpha)$ for a given axiom α are indeed weaker than α with respect to the reference ontology $\mathcal{O}^{\text{ref}}$.

Lemma 3.2. *For every $\mathcal{SROIQ}$ axiom α, if $\alpha' \in g_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(\alpha)$, then $\alpha \models_{\mathcal{O}^{\text{ref}}} \alpha'$*

Proof. We will handle each type of axiom separately.

- If $\alpha = C \sqsubseteq D$, suppose $\alpha' = C' \sqsubseteq D'$. From Lemma 3.1.2 we know that $C' \sqsubseteq_{\mathcal{O}^{\text{ref}}} C$ and $D \sqsubseteq_{\mathcal{O}^{\text{ref}}} D'$. By transitivity of subsumption, we conclude that $C \sqsubseteq D \models_{\mathcal{O}^{\text{ref}}} C' \sqsubseteq D'$.

- If $\alpha = C(a)$, suppose $\alpha' = C'(a)$. From Lemma 3.1.2 we know that $C \sqsubseteq_{\mathcal{O}^{\text{ref}}} C'$. Given any model I of $\mathcal{O}^{\text{ref}} \cup \{\alpha\}$, $a^I \in C^I$. Since $C^I \subseteq C'^I$ in every model of $\mathcal{O}^{\text{ref}}$, $a^I \in C'^I$. We conclude that $C(a) \models_{\mathcal{O}^{\text{ref}}} C'(a)$.

- If $\alpha = R(a,b)$, suppose $\alpha' = R'(a,b)$. From Lemma 3.1.2 we know that $R \sqsubseteq_{\mathcal{O}^{\text{ref}}} R'$. Given any model I of $\mathcal{O}^{\text{ref}} \cup \{\alpha\}$, $\langle a^I, b^I \rangle \in R^I$. Since $R^I \subseteq R'^I$ in every model of $\mathcal{O}^{\text{ref}}$, $\langle a^I, b^I \rangle \in R'^I$. We conclude that $R(a,b) \models_{\mathcal{O}^{\text{ref}}} R'(a,b)$.

- If $\alpha = \neg R(a,b)$, suppose $\alpha' = R'(a,b)$. From Lemma 3.1.2 we know that $R' \sqsubseteq_{\mathcal{O}^{\text{ref}}} R$. Given any model I of $\mathcal{O}^{\text{ref}} \cup \{\alpha\}$, $\langle a^I, b^I \rangle \notin R^I$. Since $R'^I \subseteq R^I$ in every model of $\mathcal{O}^{\text{ref}}$, $\langle a^I, b^I \rangle \notin R'^I$. We conclude that $\neg R(a,b) \models_{\mathcal{O}^{\text{ref}}} \neg R'(a,b)$.

- If $\alpha = \text{disjoint}(R, S)$, suppose $\alpha' = \text{disjoint}(R', S')$. From Lemma 3.1.2 we know that $R' \sqsubseteq_{\mathcal{O}^{\text{ref}}} R$ and $S' \sqsubseteq_{\mathcal{O}^{\text{ref}}} S$. Given any model I of $\mathcal{O}^{\text{ref}} \cup \{\alpha\}$, $R^I \cap S^I = \emptyset$. Since $R'^I \subseteq R^I$ and $S'^I \subseteq S^I$ in every model of $\mathcal{O}^{\text{ref}}$, $R'^I \cap S'^I = \emptyset$. We conclude that $\text{disjoint}(R, S) \models_{\mathcal{O}^{\text{ref}}} \text{disjoint}(R', S')$.
- If $\alpha = S_1 \circ \cdots \circ S_n \sqsubseteq R$, suppose $\alpha' = S'_1 \circ \cdots \circ S'_n \sqsubseteq R'$. From Lemma 3.1.2 we know that $R \sqsubseteq_{\mathcal{O}^{\text{ref}}} R'$ and $S'_i \sqsubseteq_{\mathcal{O}^{\text{ref}}} S_i$ for $i = 1, \ldots, n$. Given any model I of $\mathcal{O}^{\text{ref}} \cup \{\alpha\}$, $S_1^I \circ \cdots \circ S_n^I \subseteq R^I$. Since $R^I \subseteq R'^I$ and $S_i'^I \subseteq S_i^I$ for $i = 1, \ldots, n$ in every model of $\mathcal{O}^{\text{ref}}$, $S_1'^I \circ \cdots \circ S_n'^I \subseteq R'^I$. We conclude that $S_1 \circ \cdots \circ S_n \sqsubseteq R \models_{\mathcal{O}^{\text{ref}}} S'_1 \circ \cdots \circ S'_n \sqsubseteq R'$.

□

Clearly, replacing an axiom in an ontology with one of its weakenings cannot reduce the number of models of the ontology. However, for weakening to be useful in practice, we must additionally show that adding the weakened axioms to the ontology will not violate any of the global constraints that ensure the decidability of $\mathcal{SROIQ}$. To do this, we show first that all roles that are simple in $\mathcal{O}^{\text{full}}$ are also simple in the ontology obtained by adding the weakened axioms.

Lemma 3.3. *Let $\mathcal{O}$ be an ontology such that all simple roles of $\mathcal{O}^{\text{full}}$ are also simple in $\mathcal{O}$. For every axiom $\alpha \in \mathcal{O}$ and role R, if $\alpha' \in g_{\mathcal{O}^{\text{ref}}, \mathcal{O}^{\text{full}}}(\alpha)$ and R simple in $\mathcal{O}$, then R is simple in $\mathcal{O} \cup \{\alpha'\}$.*

Proof. (*Sketch*) For the addition to change the simplicity of any role, it must be that α' is a RIA that has some role R as the super role and is either complex or for which the sub role is non-simple. Assume, by contradiction, that R is a simple role in $\mathcal{O}$ and non-simple in $\mathcal{O} \cup \{\alpha'\}$. Since R is simple in $\mathcal{O}$ it is not the universal role, does not appear as the super role in any complex RIA of $\mathcal{O}$, and neither on the right-hand side of a simple RIA in $\mathcal{O}$ where the sub role is non-simple. If α' is a complex RIA then, by definition of the weakening operator, α must be a complex RIA and R must also be the super role in α, making it non-simple in $\mathcal{O}$, which contradicts our assumption. Similarly, if α' is a simple RIA with a non-simple role as the sub role, the sub role of α must be equal to that of α' because the refinement operators return only roles simple in $\mathcal{O}^{\text{full}}$, and those are also simple in $\mathcal{O}$. Further, since the super role of a RIA is only refined if the sub role is simple, $\alpha' = \alpha$, which means that R is non-simple in $\mathcal{O}$, which contradicts the assumptions. It follows that such a role R does not exist. □

Note that removal of axioms will never cause a role that was previously simple to become non-simple. Hence, all roles simple in $\mathcal{O}$ are also simple in $\mathcal{O} \setminus \{\alpha\} \cup \{\alpha'\}$. Further, it should be noted that repeated replacement of axioms with weakened axioms keeps simple roles simple. This is an important observation since repeated weakening is required for the proposed ontology repair algorithms using axiom weakening.

Lemma 3.4. *Let $\mathcal{O}$ be an ontology such that all simple roles of $\mathcal{O}^{\text{full}}$ are also simple in $\mathcal{O}$. For every axiom $\alpha \in \mathcal{O}$, if $\alpha' \in g_{\mathcal{O}^{\text{ref}}, \mathcal{O}^{\text{full}}}(\alpha)$ and the RBox of $\mathcal{O}$ is regular, then the RBox of $\mathcal{O} \cup \{\alpha'\}$ is also regular.*

Proof. (*Sketch*) Let us first argue that if there exists a preorder that satisfies the constraints necessary for checking regularity, then there exists $\preceq$ such that $S_1 \preceq S_2$, $S \preceq R$ and $R \not\preceq S$ for all simple roles S, S_1, S_2 and non-simple roles R. Firstly, $S_1 \not\preceq S_2$ and $S \not\preceq R$ cannot be required, because the absence of a tuple is only required for complex RIAs, where the super role must not be a predecessor of the roles

on the left-hand side. Since S_1 and S are simple, they do not appear as the super role in a complex RIA. Similarly, $R \preceq S$ cannot be required. Since S is simple and R non-simple, it cannot be required directly through an axiom of the form $R \sqsubseteq S$. By induction, it cannot be required through transitivity, since $R \preceq T$ and $T \preceq S$ would have to be required. If T is simple, $R \preceq T$ cannot be required, and if T is non-simple, $T \preceq S$ cannot be required.

Since $\mathcal{O}$ has a regular RBox, there exists such a $\preceq$ for $\mathcal{O}$. We will show that $\preceq$ is also a witness for regularity of $\mathcal{O} \cup \{\alpha'\}$. All RIAs in $\mathcal{O}$ are of one of the allowed forms for $\preceq$. It is therefore sufficient to verify that α' has one of the allowed forms. If $\alpha' = \alpha$ or α' is not a RIA, it does not affect the regularity. Otherwise, if α' is a simple RIA $S \sqsubseteq R$, then by definition of the weakening operator, S is simple in $\mathcal{O}^{\text{full}}$, and therefore also in $\mathcal{O}$. Given that S is simple, $S \preceq R$ holds for simple and non-simple R by our choice of $\preceq$. If α' is a complex RIA $S'_1 \circ \cdots \circ S'_n \sqsubseteq R$, then α is also a complex RIA $S_1 \circ \cdots \circ S_n \sqsubseteq R$ and R is non-simple in $\mathcal{O}$. If $S_i \preceq R$ and $S_i \not\preceq R$, then so will $S'_i \preceq R$ and $R \not\preceq S'_i$, either because $S_i = S'_i$ or because S'_i is simple and R is non-simple. Since $\mathcal{O}$ has a regular RBox, the only case in which $R \preceq S_i$ is if $S_i = R$. In this case, $S'_i \preceq R$ and $R \not\preceq S'_i$ will still hold if $S_i \neq R$. If $S_i = R$, either $i = 0$ or $i = n$ which is allowed. The only delicate case is if $\alpha = R \circ R \sqsubseteq R$, which will result in either $\alpha' = S'_1 \circ R \sqsubseteq R$ or $R \circ S'_2 \sqsubseteq R$, both of which are valid. □

As for the invariant on simple roles, also regularity is maintained by repeated replacement of axioms with weakenings. With the help of Lemma 3.3 and Lemma 3.4 we can now sketch a proof showing that adding weakened axioms to a $\mathcal{SROIQ}$ ontology will yield another valid $\mathcal{SROIQ}$ ontology.

Theorem 3.1. *Given that $\mathcal{O}^{\text{ref}}$ and $\mathcal{O}^{\text{full}}$ are valid $\mathcal{SROIQ}$ ontologies. For every axiom $\alpha \in \mathcal{O}^{\text{full}}$, if $\alpha' \in g_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(\alpha)$, then $\mathcal{O}^{\text{full}} \cup \{\alpha'\}$ is a valid $\mathcal{SROIQ}$ ontology.*

Proof. (*Sketch*) It has been established already in Lemma 3.4, that the regularity of the RBox will be preserved. It is guaranteed by Lemma 3.3 that all roles that were simple before addition, are still simple afterwards. Therefore, all usages of roles in axioms and concepts that were not touched by the refinement do not pose a problem. The condition that the upward and downward covers of a role contain only roles that are simple in $\mathcal{O}^{\text{full}}$ (and therefore by Lemma 3.3 also in $\mathcal{O}^{\text{full}} \cup \{\alpha'\}$) forces that every refinement of a role is simple. This restriction to simple roles guarantees that no non-simple role may be used in disjoint role axioms or the scope of cardinality and self constraints. □

Chapter 4

Implementation

The refinement and axiom weakening operators have previously been implemented for $\mathcal{ALC}$ in [64]. Based on this, the implementation has been extended to cover the full range of $\mathcal{SROIQ}$ axioms and concepts.[1] The refinement and axiom weakening operators for $\mathcal{SROIQ}$ have been implemented as discussed above. Further, repair algorithms using the axiom weakening operator based on the procedures already proposed in [64] and [16] have been implemented. The implementation performs weakening in OWL 2 DL [50] and is implemented in Java using the OWL API [24, 52, 47]. A plug-in for the ontology development tool Protégé has also been implemented and will be discussed in more detail.[2] The plug-in allows for manually weakening axioms and executing the automatic repair algorithms.

4.1 Implementing $\mathcal{SROIQ}$ Weakening

The implementation of the presented axiom weakening operator is based on the implementation at [63] for weakening in $\mathcal{ALC}$ as discussed in [64]. The implementation has been significantly extended using the operators and algorithms presented in this thesis. We will now give a brief summary covering some implementation details.

The implementation has been made such that different algorithms can be used easily for repairing the ontologies. Further, for the repairs based on axiom weakening, the approaches for the selection of the reference ontology or the bad axioms are easily configurable. Additionally, the implementation contains some extra flags that can be set to modify the behaviour of the refinement and axiom weakening operators. For example, the caching explained in Section 4.1.3 can optionally be disabled. Some other flags have been added to strictly ensure only axioms that conform to $\mathcal{ALC}$, $\mathcal{SROIQ}$, or the negation normal form of one of them are accepted and produced. Similarly, refinement of roles may optionally be disabled, used as defined in Definition 3.1, or can even be extended to use non-simple roles in contexts where they are allowed. In addition to the approach to weakening RIAs shown in Definition 3.3, the alternative approach using a fixed preorder discussed in Section 3.2.1 may be selected through a flag to the axiom weakening operator, allowing for more flexible weakening of RIAs. Note that the flags have been set up for the evaluation in Chapter 5 such that the weakening conforms to the definitions in Section 3.3, and to accept only $\mathcal{SROIQ}$ axioms.

[1] The source code is available at `https://github.com/rolandbernard/ontologyutils`.
[2] The Protégé plugin is available at `https://github.com/rolandbernard/protege-weakening`.

While the implementation features the extensions to the weakening operator discussed in Appendix B, to enable the evaluation of the proposed version of axiom weakening operating only on $\mathcal{SROIQ}$ concepts and axioms, the ontologies had to be normalized. To achieve this, axioms and concepts had to be modified. For TBox axioms, the OWL API is already able to do most of the work by providing a method that converts axioms into one or more GCIs. For ABox axioms, the only transformation that had to be done is converting n-ary same individual and different individuals axioms to $\frac{n(n-1)}{2}$ different binary equalities and inequalities. In the case of same individual axioms, there is also a flag that will cause the normalization to generate only $n-1$ equalities, such that one of the individuals is asserted to be equal to all the others. The same mechanisms have been implemented for RBox axioms asserting equivalence or disjointness between roles. The rest of the RBox axioms have been transformed into RIAs and GCIs. There are only two kinds of concepts that had to be specially handled. The exact cardinality constraint is transformed into a conjunction between an at-least and an at-most constraint. The has-value constraint is transformed by using an existential quantification and nominal concept. Further details on the implemented translation can be found in Appendix C.

Algorithm 4.1 RepairOntologyWeaken($\mathcal{O}$)

$\mathcal{O}^{\text{full}} \leftarrow \mathcal{O}$
$\mathcal{O}^{\text{ref}} \leftarrow$ FindConsistentSubset($\mathcal{O}$)
while $\mathcal{O}$ is inconsistent **do**
  $\phi_{\text{bad}} \leftarrow$ FindBadAxiom($\mathcal{O}$)
  $\phi_{\text{weaker}} \leftarrow$ SelectWeakerAxiom($g_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(\phi_{\text{bad}})$)
  $\mathcal{O} \leftarrow (\mathcal{O} \setminus \{\phi_{\text{bad}}\}) \cup \{\phi_{\text{weaker}}\}$
end while
Return $\mathcal{O}$

The automatic repair algorithm using axiom weakening has been implemented as depicted in Algorithm 4.1. It repeatedly selects "bad axioms" and replaces them with weaker axioms generated by means of the axiom weakening operator defined in Definition 3.3. This is repeated until the ontology becomes consistent. In the implementation used for the evaluation in this thesis, the reference ontology has been selected by randomly sampling a maximal consistent subset. The prototype does, however, also include multiple alternative implementations of FindConsistentSubset. One is to choose as reference ontology the intersection of some (or all) maximal consistent subsets, while another is to select one of the largest (in terms of the number of axioms) maximal consistent subsets. Note that the implementation will consider ontologies as the union of some static axioms $\mathcal{O}_s$ and some refutable axioms $\mathcal{O}_r$, as defined in Section 2.4.1. This has been taken into account for the computation of maximal consistent subsets in FindBadAxiom. One reason for the split into static and refutable axioms is to keep non-logical axioms, like declarations and annotations, in every tested ontology. Especially declaration axioms must be kept, because OWL 2 DL mandates that every signature element must be declared, and otherwise some reasoners will produce incorrect results or throw errors.

The procedure FindBadAxiom may also be implemented in several different ways. For the evaluation, we consider an implementation that randomly samples some (or all) minimal inconsistent subsets of axioms $J_1, J_2, \ldots, J_k \subseteq \mathcal{O}$ and selects as the bad axiom the one occurring most frequently. The implementation contains also some

other implementations of FindBadAxiom, e.g., selecting a random axiom, selecting the axiom that appears in the least maximal consistent subsets, or selecting an axiom that does not appear in the largest (by axiom count) maximal consistent subset. The implementation of FindBadAxiom is such that it will only ever return axioms $\alpha \in \mathcal{O}_r$. To do this, minimal subsets J of $\mathcal{O}_r$ such that $J \cup \mathcal{O}_s$ is inconsistent are sampled. With this feature, a variation of the repair using axiom weakening has been implemented, in which the axioms of the reference ontology $\mathcal{O}^{\text{ref}}$ are not weakened during the repair. This has been done by, after computing $\mathcal{O}^{\text{ref}}$, setting $\mathcal{O}_s \leftarrow \mathcal{O}_s \cup \mathcal{O}^{\text{ref}}$ and $\mathcal{O}_r \leftarrow \mathcal{O}_r \setminus \mathcal{O}^{\text{ref}}$. This variation has been evaluated in Section 5.1.

4.1.1 Performance

The prototype implementation uses the OWL API [24, 52] to represent axioms in memory and interface with off-the-shelf reasoners. The reasoners FaCT++, JFact, HermiT, and Openllet have been used in the prototype. Creating a new reasoner instance for each time an entailment is checked or consistency must be tested is however inefficient. One reason for this is that instantiating a reasoner is relatively expensive. Further, all the mentioned reasoners are, to some degree, able to keep some information computed for previous queries and exploit it to speed up subsequent ones. For these reasons, it is desirable to reuse reasoners as much as possible. To facilitate this, a class `Ontology` has been created that contains a set of axioms (split into static $\mathcal{O}_s$ and refutable $\mathcal{O}_r$) as well as a reference to a set of already initialized reasoners. This class contains the methods for checking consistency and entailment, implementing them by taking one of the initialized reasoners, updating the reasoner's internal state by adding and removing axioms to reflect the state of the ontologies axioms, and then calling the reasoners axioms. Only if no reasoner is available, will a new one be created and initialized. One thing this setup allows for is the creation of multiple ontologies that point to the same set of reasoners. To achieve this, it is recorded which ontologies use the reasoners, and if they are no longer needed, they can be disposed of to free up global resources.

To improve performance, particularly for running the test cases, running work in parallel has been considered. One problem with this approach is that all reasoners are linked to a global `OWLOntologyManager` object. There is a concurrent version of this object, but not all reasoners are implemented in a way to make concurrent access to this object possible. The FaCT++ reasoner [1] had a significantly severe problem, where it would listen to and apply all changes made to any ontology in the ontology manager. Since this manager can, however, contain multiple ontologies, this lead to wrong reasoning results. These issues have been resolved in the slightly modified version of the FaCT++ reasoner used for this thesis.[3] This version had some further modifications, like fixing an issue with the handling of annotated axioms and adding automatic loading of native dependencies, something for which previously an environment variable had to be set pointing at the dynamic library. After these changes, and some additional modifications to the prototype, the tests can now be run concurrently. Further, the prototype contains some experiments that use multiple threads, sharing the same axiom weakening operator and cache, to compute multiple weakenings in parallel.

[3]The version of FaCT++ used for this thesis is available at `https://github.com/rolandbernard/factplusplus`

4.1.2 Computing Minimal Subsets

For some operations, maximal consistent subsets and minimal inconsistent subsets of an ontology $\mathcal{O}$ must be computed. As has been observed in [46] for the case of propositional logic, these problems can both be reduced to the problem of finding a minimal subset over some monotone predicate. This is also true for DLs. A predicate p is monotone if for every two sets A and B where $A \subseteq B$, $p(A) \implies p(B)$. Given a monotone predicate p and a set A, the problem is to find a subset J of A such that $p(J)$ and for all subsets J' of J, $p(J')$ does not hold. The problem of finding a minimal inconsistent subset can trivially be reduced to this problem by setting $A = \mathcal{O}_r$ and $p_{\text{MIS}}(A) \iff \mathcal{O}_s \cup A \models \top \sqsubseteq \bot$. Instead of finding the maximal consistent subset, we can find the minimal subset such that removing it from the full ontology results in a consistent ontology. That is, we solve the minimal subset problem over the set $A = \mathcal{O}_r$ using the predicate $p_{\text{MCS}}(A) \iff \mathcal{O}_s \cup \mathcal{O}_r \setminus A \not\models \top \sqsubseteq \bot$. Removing this minimal subset will yield a maximal consistent subset since adding any other axiom is equivalent to removing fewer axioms, and since the set is minimal, there is no way to remove fewer axioms and still satisfy the predicate.

Algorithm 4.2 SimpleFindMinimalSubset(A, p)

```
A' ← A
for a ∈ A do
    if p(A' \ {a}) then
        A' ← A' \ {a}
    end if
end for
Return A'
```

One simple way to compute a single minimal subset with respect to p is by starting with the complete set. Then one goes through all the elements and tries to remove them. If after removal the predicate still holds, the element is removed, otherwise, it is kept. This algorithm is listed in Algorithm 4.2. After having tested and possibly removed all elements, the remaining elements form a minimal subset. To see this, assume we end up with a set A' and there exists an element $a \in A'$ for which $p(A' \setminus \{a\})$ is true. Since all elements in A have already been tested once, $p(A'' \setminus \{a\})$ was false for some set $A'' \supseteq A'$. However, this means $A' \setminus \{a\} \subseteq A'' \setminus \{a\}$ and since p is monotone and $p(A' \setminus \{a\})$ is true, $p(A'' \setminus \{a\})$ must also be true. This is a contradiction since if $p(A'' \setminus \{a\})$ had been true we would have removed a from the set.

This same idea can be made somewhat more efficient in cases where the minimal subsets are only relatively small compared to the complete set. This is often the case for finding minimal inconsistent subsets or minimal correction subsets. For this, the progression-based algorithm presented in [46] has been implemented as depicted in Algorithm 4.3. The algorithm works by trying to remove more than one element at a time, and if that fails, binary search is used to locate the first element which must be included in the minimal set. To compute multiple (but not necessarily all) minimal sets, the MergeXplain algorithm proposed in [61] has also been implemented, but was not used during the evaluation.

Similarly, an algorithm for finding all minimal subsets has been implemented as shown in Algorithm 4.4. The algorithm is based on the hitting-set-tree algorithm [53] and is based on the approach proposed in [33]. The idea is to use a depth-first search on the hitting-set-tree. In this tree, every node is labelled with a minimal subset,

Algorithm 4.3 FINDMINIMALSUBSET(A, p)

```
Choose some permutation a_1, a_2, ··· , a_n of A
A' ← ∅,  i ← 1,  s ← 1
while i ≤ n do
    if p(A' ∪ {a_{i+s}, a_{i+s+1}, ··· , a_n}) then
        i ← i + s,  s ← 2s
    else
        l ← i,  r ← i + s − 1
        while l < r do
            m ← ⌈(l+r)/2⌉
            if p(A' ∪ {a_m, a_{m+1}, ··· , a_n}) then
                l ← m
            else
                r ← m − 1
            end if
        end while
        A' ← A' ∪ {a_l},  i ← l + 1,  s ← 1
    end if
end while
Return A'
```

and every edge is labelled with an element of the minimal subset in the parent. The elements labelling the edges on the path from the root to a node are removed from the complete set before finding a minimal subset with which to label the node. To find the minimal subset, the procedure in FINDMINIMALSUBSET can be used. If no such minimal subset exists, then the search in the current branch is finished, because no subset will contain any minimal subsets either. Minimal subsets may be reused to label multiple nodes, making the search more efficient. Additionally, early path pruning can be performed to avoid testing the same subsets multiple times.

4.1.3 Caching for Faster Repairs

The implementation will spend most of its time during repairs calling the reasoner to compute consistency and entailment queries. In order to reduce the number of calls that must be made and to accelerate the computation of upward and downward cover sets, some caching has been added. Since upward and downward covers must potentially be computed many times for a single application of the refinement or axiom weakening operators, it is an obvious point to optimize. Further, since the reference ontology and full ontology have been chosen to remain fixed for the duration of the repair, the upward and downward cover functions remain constant. When following directly the definition of cover sets presented in Definition 3.1, one will compute numerous subsumptions, many of which will be the same across multiple cover computations. This is therefore the aspect that caching has been applied to. The reasoners used in the implementation already have the ability to partially reuse previous results when computing queries. We, therefore, made sure to reuse the same reasoner instances wherever possible, to exploit this feature.

Rather than rely solely on the internal reuse of the reasoner or apply only a simple cache, however, we found it worthwhile to create a cache for subsumption, in which extra information is also inferred from the transitivity of subsumption after

Algorithm 4.4 FindAllMinimalSubsets(A, p)

```
Globals  B ← ∅,  M ← ∅    (initialized once)
if ∀b ∈ B : A ⊈ b then
    if ∃m' ∈ M : m' ⊆ A then
        m ← m'
    else if p(A) then
        m ← FindMinimalSubset(A, p)
        M ← M ∪ {m}
    else
        m ← ∅
    end if
    for a ∈ m do
        FindAllMinimalSubsets(A \ {a}, p)
    end for
    B ← B ∪ {A}
end if
M contains the set of all minimal subsets after the DFS terminates.
```

each query to the reasoner. This is in effect similar to the technique presented in [62] for creating taxonomies, but applied also to complex concepts. Additionally, some basic rules were added to compute subsumption involving conjunctions and disjunctions, by using the results computed for the conjuncts or disjuncts of the expression. Another difference, with respect to the computation of taxonomies, is that subsumptions are computed only when requested. While pre-computing the complete relation over the subconcepts would allow for ordering of the reasoner queries to maximize information gain, like one may do when computing taxonomies, this turned out to be disadvantageous, since in general, a single repair will not require all results.

It follows a brief description of the algorithm used for caching subsumptions listed in Algorithm 4.5. We keep three global variables, the set of all elements that have already been encountered at least once S, the set of know tuples K, and the set of possible tuples P. When querying a subsumption, it is first ensured that if one or both of the elements has not yet been seen before, the appropriate tuples are inserted into K and P. Then it is checked whether the result can be determined using the known information, and if not, the procedure TestSubsumption is used to compute the correct result. For every new negative result, the possible tuples are reduced and for every positive result, known tuples are added, and possible tuples may also be removed. The algorithm is based on the two implications

$$C \sqsubseteq_{\mathcal{O}} D \text{ and } D \sqsubseteq_{\mathcal{O}} E \implies C \sqsubseteq_{\mathcal{O}} E \text{ , and}$$
$$C \not\sqsubseteq_{\mathcal{O}} D \text{ and } C \sqsubseteq_{\mathcal{O}} E \text{ and } F \sqsubseteq_{\mathcal{O}} D \implies E \not\sqsubseteq_{\mathcal{O}} F \text{ .}$$

The first implication follows trivially from the fact that subsumption between concepts or roles is transitive. The second also follows from transitivity. It can be observed that, if $C \sqsubseteq_{\mathcal{O}} E$ and $F \sqsubseteq_{\mathcal{O}} D$, if $E \sqsubseteq_{\mathcal{O}} F$ were to hold, then $C \sqsubseteq_{\mathcal{O}} D$ would also have to hold by transitivity. Separate instances of S, K and P are used for the computation of concept and role subsumptions.

For actually computing subsumptions between roles, TestSubsumption is a simple call to the reasoner. However, for computing subsumption between concepts, one further step is used. It is applied immediately before possibly calling the reasoner.

Algorithm 4.5 CachedTestSubsumption(C, D)

Globals $S \leftarrow \emptyset$, $K \leftarrow \emptyset$, $P \leftarrow \emptyset$ (initialized once)
for $X \in \{C, D\}$ where $X \notin S$ **do**
  $S \leftarrow S \cup \{X\}$
  $K \leftarrow K \cup \{\langle X, X\rangle\}$
  $P \leftarrow P \cup (S \times \{X\}) \cup (\{X\} \times S)$
end for
if $\langle C, D\rangle \in K$ **then**
  $R \leftarrow \mathit{True}$
else if $\langle C, D\rangle \notin P$ **then**
  $R \leftarrow \mathit{False}$
else
  $R \leftarrow$ TestSubsumption(C, D)
  if R **then**
    $K \leftarrow K \cup \{\langle C', D'\rangle \mid \langle C', C\rangle \in K \text{ and } \langle D, D'\rangle \in K\}$
    $P \leftarrow P \setminus \{\langle D, D'\rangle \mid \langle D', C'\rangle \in K \text{ and } \langle C, C'\rangle \notin P\}$
    $P \leftarrow P \setminus \{\langle C', C\rangle \mid \langle D', C'\rangle \in K \text{ and } \langle D', D\rangle \notin P\}$
  else
    $P \leftarrow P \setminus \{\langle C', D'\rangle \mid \langle C, C'\rangle \in K \text{ and } \langle D', D\rangle \in K\}$
  end if
end if
Return R

If any of the following rules can be computed using only the information in S, K, and P, the reasoner is not called, and the result returned from TestSubsumption. Otherwise, the reasoner is used as usual to compute the subsumption.

$$
\begin{aligned}
C_i \not\sqsubseteq_{\mathcal{O}} D \text{ for some } i \in \{1, \ldots, n\} &\implies C_1 \sqcup \cdots \sqcup C_n \not\sqsubseteq_{\mathcal{O}} D \\
C_i \sqsubseteq_{\mathcal{O}} D \text{ for all } i \in \{1, \ldots, n\} &\implies C_1 \sqcup \cdots \sqcup C_n \sqsubseteq_{\mathcal{O}} D \\
C \not\sqsubseteq_{\mathcal{O}} D_i \text{ for some } i \in \{1, \ldots, n\} &\implies C \not\sqsubseteq_{\mathcal{O}} D_1 \sqcap \cdots \sqcap D_n \\
C \sqsubseteq_{\mathcal{O}} D_i \text{ for all } i \in \{1, \ldots, n\} &\implies C \sqsubseteq_{\mathcal{O}} D_1 \sqcap \cdots \sqcap D_n \\
C_i \sqsubseteq_{\mathcal{O}} D \text{ for some } i \in \{1, \ldots, n\} &\implies C_1 \sqcap \cdots \sqcap C_n \sqsubseteq_{\mathcal{O}} D \\
C \sqsubseteq_{\mathcal{O}} D_i \text{ for some } i \in \{1, \ldots, n\} &\implies C \sqsubseteq_{\mathcal{O}} D_1 \sqcup \cdots \sqcup D_n
\end{aligned}
$$

4.1.4 Finding Best Repairs

The implementation contains also some approaches for finding the "best repair" based on some quality function given to the algorithm. None of these algorithms were however evaluated since they are significantly slower than the others, needing to compute not only multiple repairs, but also the quality estimation for each of them. The simplest algorithm implemented for this purpose simply computes some (or all) maximal consistent subsets and selects the one that performs the based on the given function. Another is to compute multiple repairs using axiom weakening, following the algorithm in Algorithm 4.1, and select the one that has the highest quality based on the given function. A more sophisticated approach that was also realized is to apply a Monte-Carlo search [17] based algorithm to explore the tree formed by weakening. In

this case, both the possible choices of bad axioms and the selection of the weakening are explored using search.

4.2 Axiom Weakening for Protégé

The Protégé plugin reuses much of the code written for the above-mentioned implementation. For the implementation, the information in the developer documentation [2] has been consulted. Additionally, the implementation of other plugins [57, 66] has been used as a basis for this plugin. It would have been desirable to share the same codebase between the two implementations, but some choices made for the prototype implementation were incompatible with its later use in the plugin. While Protégé is also using the OWL API, it is not using the latest version. Since for the implementation discussed in Section 4.1 the latest OWL API version was used, and the two are not compatible, some changes had to be made. Further, the prototype has as dependencies the different OWL reasoners, this is a problem because those might not be included in Protégé, and bundling them with the plugin might cause conflicts if they are actually in Protégé already. For this reason, the implementation has been adapted to use the reasoner that is currently selected by the user in Protégé. Additionally, but less impactful, is also that Protégé uses an older version of Java than what has been used for the prototype implementation build for the evaluation. Three different ways to interact with the axiom weakening methods have been implemented in this plugin, we will show now each of them in turn.

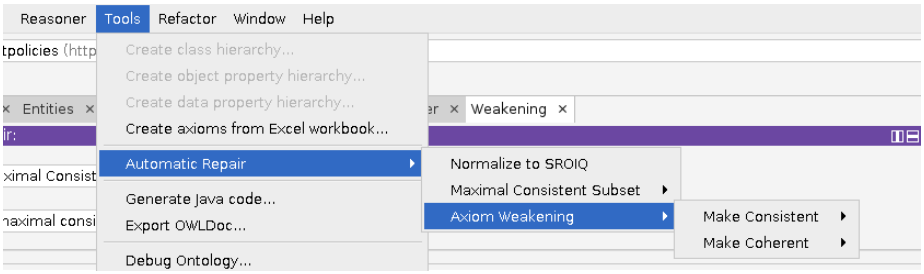


Figure 4.1: Screenshot of the Protégé menu items added by the plugin.

To quickly apply the normalization or automatic repair algorithms, some menu items have been added by the plugin under "Tools". The first menu item, "Normalize to SROIQ", allows the quick application of the normalization, described in Appendix C. Then there are two different repair algorithms for which menu entries have been added. The first, used by "Maximal Consistent Subset" selects as the repair a randomly selected maximal consistent subset. The second, used with "Axiom Weakening", is based on the algorithm listed in Algorithm 4.1. For each of these two algorithms, the user may choose between making the ontology only consistent or making it also coherent. The repair for making the ontology coherent works analogously to the one for making the ontology consistent, except that the tests for consistency of the ontology are replaced with tests for coherence.

The second method of interacting with the proposed axiom weakening repair algorithms is through a more detailed view added by the plugin. The user can select both the algorithm, e.g., by removal, using a maximally consistent set, using axiom weakening, etc., and the goal of the repair, i.e., consistency or coherence. Further, for

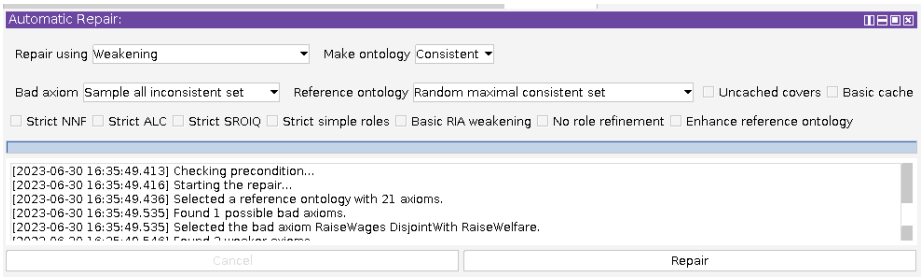


Figure 4.2: Screenshot of the Protégé view added by the plugin, allowing for configuring and applying repairs algorithms.

some repair methods, options are made available. For the repair by weakening, all the flags and variations discussed in Section 4.1 can be configured, as can be seen in Fig. 4.2. Below the section for configuring the repair algorithm, a progress indicator and log have been placed. For long-running repairs, this is a valuable addition so that the user can see that the repair is still running. Another important feature that has been added specifically for the plugin is the ability to interrupt the repairs. To facilitate this functionality, the repair is started in a separate thread, and before every reasoner call, the implementation checks whether the thread has been interrupted, and if it has the repair is terminated by throwing an exception.

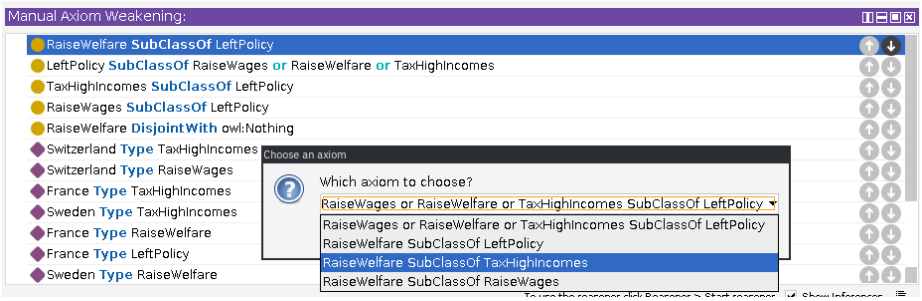


Figure 4.3: Screenshot of the Protégé view added by the plugin, allowing for manually selecting axioms and weakening them.

Lastly, a view has been added that allows users to manually select how they want to weaken axioms. The view, depicted in Fig. 4.3, shows a list of all axioms. The axioms are sorted by how frequently they appear in some minimal inconsistent subsets that are randomly sampled. The axioms that appear the most often are shown first, such that the axioms that are shown at the top of the list are the ones that would be selected by the automatic repair algorithm. The two buttons next to the axioms with the upward and downward arrows can then be used to, respectively, show weaker or stronger axioms. The user may choose one of these refined axioms to replace the original axiom.

Chapter 5

Experiments and Evaluation

The evaluation of the proposed refinement and axiom weakening operators has been carried out using the implementation described in Section 4.1 using different ontologies from BioPortal [67, 12]. Additionally, the pizza ontology [18] was included in the testing. The chosen ontologies are of different sizes and use a varying amount of expressive features. Some characteristics of the used ontologies are shown in Table 5.1. On average, they contain about 289 axioms, 73 concept names, 29 role names, and 168 subconcepts. Some additional evaluation results that did not fit into this chapter can be found in Appendix F.

Abbreviation	Name	Expressivity	Axioms	Concepts	Roles	Subcon.
admin	Nurse Administrator	$\mathcal{ALCHOIF}$	229	42	29	144
ahso	Animal Health Surveillance Ontology	$\mathcal{ALCRIF}$	166	38	31	104
cdao	Comparative Data Analysis Ontology	$\mathcal{ALCROIQ}$	437	132	68	375
cdpeo	Chronic Disease Patient Education	$\mathcal{ALCHF}$	422	41	31	170
covid19-ibo	Covid-19 Impact on Banking Ontology	$\mathcal{ALCH}$	288	160	33	227
ecp	Electronic Care Plan	$\mathcal{ALCRQ}$	127	33	17	99
emo	Enzyme Mechanism Ontology	$\mathcal{ALCHQ}$	368	157	24	255
evi	Evidence Graph Ontology	$\mathcal{ALCRI}$	143	30	38	69
falls	Falls Prevention	$\mathcal{ALCH}$	79	30	20	35
fo	Fern Ontology	$\mathcal{ALCHI}$	59	31	4	46
gbm	Glioblastoma	$\mathcal{ALCIF}$	603	108	28	227
gfvo	Genomic Feature and Variation Ontology	$\mathcal{ALCH}$	332	102	30	170
koro	Knowledge Object Reference Ontology	$\mathcal{ALCHI}$	262	110	19	194
lico	Liver Case Ontology	$\mathcal{ALCHQ}$	366	93	36	230
mamo	Mathematical Modelling Ontology	$\mathcal{ALCR}$	229	107	3	154
mpio	Minimum PDDI Information Ontology	$\mathcal{ALCH}$	38	30	14	45
pizza	Pizza Ontology	$\mathcal{SHOIN}$	1131	100	8	376
provo	Provenance Ontology	$\mathcal{ALCRIN}$	170	31	42	128
qudt	Quantities, Units, Dimensions, and Types	$\mathcal{SHOIQ}$	293	74	73	177
trans	Nurse Transitional	$\mathcal{ALCROIF}$	244	44	22	123
triage	Nurse triage	$\mathcal{ALCHF}$	132	33	29	129
vio	Vaccine Investigation Ontology	$\mathcal{ALCRI}$	249	81	44	235

Table 5.1: The ontologies used for evaluation. The number of axioms, concept names, role names, and subconcepts are taken after applying preprocessing.

Since the ontologies use OWL 2, the axioms and concepts do not map directly to $\mathcal{SROIQ}$. It is possible to directly apply axiom weakening to OWL 2 DL ontologies, a description of such a scheme and why it might be beneficial can be found in Appendix B. However, to follow the definitions laid out in Section 3.3, the OWL 2 DL axioms have been translated to $\mathcal{SROIQ}$ axioms. A detailed description of the mapping between OWL 2 DL and $\mathcal{SROIQ}$ used for this evaluation can be found in Appendix C. While some axioms have a direct translation, others have been replaced

by a set of axioms that together are equivalent. During the preprocessing, we further removed annotation axioms and axioms related to data properties and any axiom that caused a violation of the OWL 2 DL profile, as our weakening operator does not handle them.

Claims about runtime should generally only be used for relative comparisons and rough estimates. While they will obviously vary greatly based on hardware choices, all evaluations here have been performed on an Intel Coffee Lake system running at around 4GHz for the duration of the experiment. Unless otherwise indicated, the FaCT++ reasoner [65, 1] was used for evaluation.

5.1 The Quality of Repairs

In order to evaluate the proposed approach of axiom weakening, specifically for its application in the context of automatic repair of ontologies, we need a way to compare the quality of different possible repairs. As has already been discussed in [64], the problem of deciding which of two possible repaired ontologies $\mathcal{O}_1$ or $\mathcal{O}_2$ is preferable is not generally well-defined. As such, this thesis will use the same measure defined for the experimental evaluation in [64]. The main idea is to use the size of the *inferred class hierarchy* to evaluate the amount of "information" a repair can retain.

Definition 5.1. The *inferred class hierarchy* of an ontology $\mathcal{O}$ is given by

$$\mathrm{Inf}(\mathcal{O}) = \{A \sqsubseteq B \mid A, B \in N_C \text{ and } A \sqsubseteq_{\mathcal{O}} B\} \ .$$

To compare the relative amount of "information" between two ontologies, we compare them based on their differences in the inferred class hierarchy. We use for this purpose the *inferable information content* (IIC) as defined in [64].

Definition 5.2. The *inferable information content* of an ontology $\mathcal{O}_1$ with respect to another ontology $\mathcal{O}_2$ is given by

$$\mathrm{IIC}(\mathcal{O}_1, \mathcal{O}_2) = \frac{|\mathrm{Inf}(\mathcal{O}_1) \setminus \mathrm{Inf}(\mathcal{O}_2)|}{|\mathrm{Inf}(\mathcal{O}_1) \setminus \mathrm{Inf}(\mathcal{O}_2)| + |\mathrm{Inf}(\mathcal{O}_2) \setminus \mathrm{Inf}(\mathcal{O}_1)|} \ ,$$

where $|X|$ is the cardinality of the set X.

The IIC of an ontology $\mathcal{O}_1$ with respect to a second ontology $\mathcal{O}_2$, written $\mathrm{IIC}(\mathcal{O}_1, \mathcal{O}_2)$, is a number between 0 and 1. A value closer to 1 indicates that $\mathcal{O}_1$ contains more "information" than $\mathcal{O}_2$, while a value towards 0 indicates the opposite. Inferred hierarchy axioms common to both ontologies do not influence the results, and if $\mathcal{O}_2$ is entailed by $\mathcal{O}_1$, then the IIC of $\mathcal{O}_1$ with respect to $\mathcal{O}_2$ is always 1. If both $\mathcal{O}_1$ and $\mathcal{O}_2$ have some inferences that are not shared by the other ontology, the IIC will be strictly between 0 and 1. Note that if the cardinality of the inferred class hierarchy is larger for one ontology, then it will also be the one preferred by the IIC. Some weaknesses of this measure when it comes to evaluating repairs, like the fact that only atomic concepts are considered, have already been discussed in [64]. For the case of repairing $\mathcal{SROIQ}$ ontologies, this is even more relevant, since the role hierarchy is entirely ignored.

The evaluation starts by first making the ontologies inconsistent, by repeatedly adding axioms to the ontology until it is no longer consistent. The newly added axioms were generated by strengthening randomly selected axioms of the ontology.

Axioms are strengthened by applying an axiom strengthening operator, that is equivalent to the axiom weakening operator in Definition 3.3, except for swapping the generalization and specialization operators and not using $\bot \sqsubseteq \top$ to remove axioms. An axiom α' is stronger than an axiom α with respect to the ontology $\mathcal{O}$ if $\alpha' \models_{\mathcal{O}} \alpha$. Note that this is only a useful characterization if $\alpha \notin \mathcal{O}$. It was ensured that the added axioms on their own were not inconsistent. After making the ontology inconsistent, it was repaired using different algorithms. Each inconsistent ontology was repaired once with the repair algorithm shown in Algorithm 4.1 and once using Algorithm 5.1. As can be seen in Example 2.11, choosing a maximal consistent subset can sometimes lead to less information loss than using the weakening based algorithm. To experimentally evaluate how good weakening performs against the maximal consistent subset based repair on average, the repair was also performed by selecting a randomly sampled maximal consistent subset.

Algorithm 5.1 RepairOntologyRemove($\mathcal{O}$)

while $\mathcal{O}$ is inconsistent **do**
  $\phi_{\text{bad}} \leftarrow$ FindBadAxiom($\mathcal{O}$)
  $\mathcal{O} \leftarrow \mathcal{O} \setminus \{\phi_{\text{bad}}\}$
end while
Return $\mathcal{O}$

This process, both making the ontology inconsistent and repairing it, was repeated one hundred times for each ontology, and the IIC was computed between the results obtained using the different repair methods. The evaluation was performed using a randomly sampled maximal consistent subset as the reference ontology and by randomly sampling 16 minimal inconsistent subsets during the selection of bad axioms in FindBadAxiom. To obtain comparable results, both RepairOntologyRemove and RepairOntologyWeakening use the same implementation of FindBadAxiom. While the utilized OWL 2 DL reasoners are all highly optimized, they exhibit undesirable performance in some edge cases. Mostly they are fast to perform reasoning tasks like checking for consistency or entailment in the selected ontologies. However, when performance pitfalls are encountered, they may require significantly more time, making the computations required for repair unreasonably slow. For this reason, a timeout of 5 minutes was placed on the repairs execution and the outputs of runs that did not complete within this time limit were discarded and replaced by new runs. The results of these experiments are listed in Table 5.2 and shown in Fig. 5.1 and Fig. 5.2.

The results of the evaluation suggest that the repair by weakening is on average about as good or better than the repair by removal. While this supports the conclusion in [64] that axiom weakening is able to retain more information than removal, the observed advantage was smaller than what has been observed in [64].

In contrast, it can be seen that the repair using weakening is not always better than choosing a random maximal consistent subset. There are ontologies for which the repair by weakening is on average significantly worse when comparing using IIC. This is, however, a somewhat unequal comparison. We have seen in Example 2.11 also a situation in which weakening performs worse when it comes to preserving information compared to choosing a maximally consistent subset. A more equal comparison, yet still not perfect since the maximal consistent subset for the reference ontology is not chosen uniformly at random, would be to a repair algorithm that starts with the reference ontology and uses axiom weakening to add more information from the

	IIC w.r.t. removal	IIC w.r.t. maximal consistent subset	
admin	0.56 [0.51, 0.60]	0.39 [0.33, 0.45]	(0.51 [0.45, 0.57])
ahso	0.57 [0.52, 0.62]	0.52 [0.47, 0.56]	(0.64 [0.59, 0.69])
cdao	0.54 [0.49, 0.61]	0.51 [0.44, 0.57]	(0.55 [0.48, 0.61])
cdpeo	0.50 [0.47, 0.53]	0.25 [0.20, 0.30]	(0.39 [0.33, 0.45])
covid19-ibo	0.71 [0.67, 0.75]	0.62 [0.58, 0.66]	(0.72 [0.68, 0.76])
ecp	0.75 [0.70, 0.80]	0.36 [0.30, 0.42]	(0.50 [0.44, 0.56])
emo	0.68 [0.63, 0.72]	0.61 [0.56, 0.65]	(0.73 [0.68, 0.77])
evi	0.51 [0.46, 0.55]	0.58 [0.53, 0.63]	(0.71 [0.66, 0.75])
falls	0.75 [0.70, 0.81]	0.50 [0.44, 0.57]	(0.60 [0.54, 0.66])
fo	0.53 [0.48, 0.58]	0.68 [0.63, 0.73]	(0.67 [0.62, 0.73])
gbm	0.59 [0.54, 0.64]	0.54 [0.49, 0.59]	(0.68 [0.63, 0.73])
gfvo	0.53 [0.47, 0.58]	0.54 [0.49, 0.59]	(0.71 [0.67, 0.75])
koro	0.53 [0.48, 0.58]	0.38 [0.32, 0.43]	(0.53 [0.47, 0.59])
lico	0.53 [0.48, 0.59]	0.51 [0.46, 0.57]	(0.63 [0.58, 0.68])
mamo	0.55 [0.49, 0.61]	0.64 [0.58, 0.69]	(0.72 [0.66, 0.77])
mpio	0.69 [0.65, 0.74]	0.70 [0.66, 0.75]	(0.70 [0.65, 0.74])
pizza	0.56 [0.51, 0.62]	0.57 [0.51, 0.63]	(0.63 [0.58, 0.69])
provo	0.51 [0.47, 0.56]	0.55 [0.51, 0.60]	(0.64 [0.60, 0.69])
qudt	0.96 [0.93, 0.98]	0.47 [0.40, 0.53]	(0.56 [0.49, 0.62])
trans	0.56 [0.51, 0.61]	0.41 [0.35, 0.47]	(0.51 [0.45, 0.57])
triage	0.51 [0.46, 0.56]	0.52 [0.47, 0.57]	(0.63 [0.57, 0.68])
vio	0.48 [0.41, 0.54]	0.50 [0.44, 0.57]	(0.57 [0.51, 0.63])
Overall	0.60 [0.58, 0.61]	0.52 [0.50, 0.53]	(0.61 [0.60, 0.63])

Table 5.2: Results of the evaluation. IIC is given as mean and 95% confidence interval in brackets. The values in parentheses are for weakening that does not weaken the axioms in the maximal consistent subset chosen as reference ontology.

remaining axioms. This has been implemented and evaluated, and the results can be seen as the orange bars in Fig. 5.2. Still, this result suggests that the heuristic used for selecting bad axioms is not reliable for preserving information, at least when measured using IIC. It has not been closely studied what causes the repair by weakening to significantly lose for some ontologies while being clearly preferred in others. Interestingly, there are also some cases in which the weakening based repair performs better against a random maximal consistent subset than against the repair by removal.

To overcome some weaknesses of the IIC, especially that it does not account for subsumptions between roles or complex concept expressions, tests were also carried out using an extended version of the inferred class hierarchy and IIC, adding to it subsumptions between subconcepts and roles.

Definition 5.3. The *extended inferred class hierarchy* of an ontology $\mathcal{O}$ is given by

$$\begin{aligned}\mathrm{Inf}^{+}(\mathcal{O}) = \{C \sqsubseteq D \mid C, D \in \mathrm{sub}(\mathcal{O}) \text{ and } C \sqsubseteq_{\mathcal{O}} D\} \\ \cup \{R \sqsubseteq S \mid R, S \in \mathcal{L}(N_R) \text{ and } R \sqsubseteq_{\mathcal{O}} S\} \ .\end{aligned}$$

The *extended inferable information content* (IIC$^+$) of an ontology $\mathcal{O}_1$ with respect to another ontology $\mathcal{O}_2$ is given by

$$\mathrm{IIC}^{+}(\mathcal{O}_1, \mathcal{O}_2) = \frac{|\mathrm{Inf}^{+}(\mathcal{O}_1) \setminus \mathrm{Inf}^{+}(\mathcal{O}_2)|}{|\mathrm{Inf}^{+}(\mathcal{O}_1) \setminus \mathrm{Inf}^{+}(\mathcal{O}_2)| + |\mathrm{Inf}^{+}(\mathcal{O}_2) \setminus \mathrm{Inf}^{+}(\mathcal{O}_1)|} \ ,$$

Of course, this new measure still only captures a small part of all consequences, but it should provide more accurate results when it comes to information about subconcepts and roles. When evaluating the experiment outcomes, we see that the values produced by IIC and IIC$^+$ are very similar. As may be expected, the repair by weakening is preferred slightly more strongly when using IIC$^+$. Figure 5.3 shows the comparison between IIC and IIC$^+$ for repair using axiom weakening w.r.t. repair by

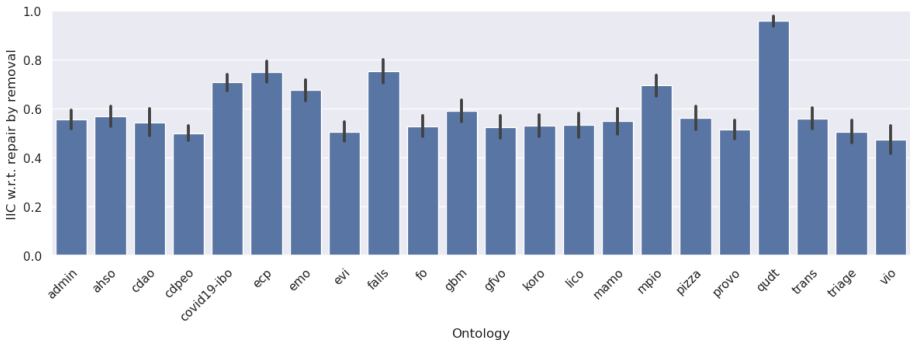


Figure 5.1: Mean IIC with respect to repair via removal per ontology. The error bars show the 95% confidence interval.

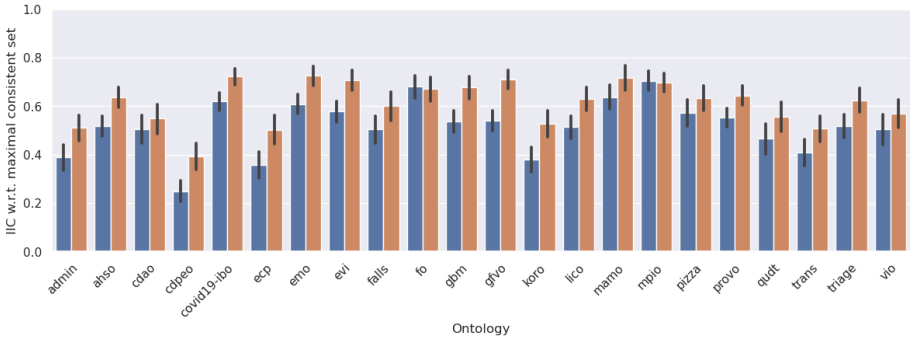


Figure 5.2: Mean IIC with respect to a random maximal consistent subset. The error bars show the 95% confidence interval. The orange bars used a variation of the repair by weakening, where axioms in the reference ontology are never weakened.

removal. The differences between IIC and IIC^+ are somewhat stronger when comparing against the repair using a random maximal consistent subset, but the differences are still not particularly noteworthy.

5.2 Caching in Cover Computations

As discussed in Section 4.1.3, to accelerate the computation of upward and downward covers, a cache was added to avoid repeated calls to the reasoner when the necessary information can already be inferred from previous computations. This section will discuss the experiments performed to evaluate the effectiveness of this cache.

Three versions of the upward and downward cover have been considered, the uncached version that calls the reasoner for every subsumption query, a version that caches only the exact queries that were already made previously, and finally the version that additionally infers some additional information from the transitivity of subsumption as listed in Algorithm 4.5. The experiment uses some of the same ontologies as have been used for the previously discussed evaluation of the repair quality. From the logical axioms of each ontology were selected, uniformly at random, one hundred groups with a fixed number of axioms. The same axioms may be selected multiple

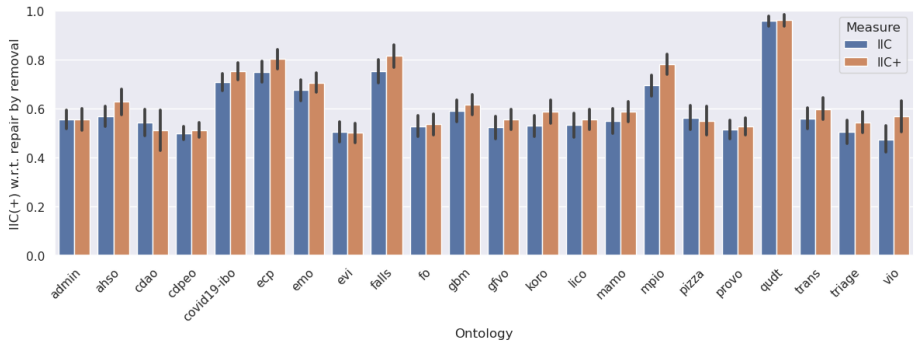


Figure 5.3: Mean IIC and IIC$^+$ with respect to repair by removal per ontology. The error bars show the 95% confidence interval.

times. Each of the groups was then tested with a separate instance of the cache. The rationale behind testing with different group sizes is that the cache will obviously have a greater impact when the same cache can be reused for more cover computations. In each test run, the axiom weakening operator was applied to each axiom and the number of reasoner calls and (real) time taken were measured. The weakening operator used the complete (after preprocessing) ontology as the reference ontology. The test has further been performed using different OWL 2 DL reasoners. The final results of the evaluation can be seen in Table 5.3 and are visualized in Fig. 5.4 and Fig. 5.5. Note that a logarithmic y-axis has been used for both plots.

	Reasoner calls per weakening								
	Full caching			Simple caching			No caching		
	1	10	100	1	10	100	1	10	100
admin	1096	222	35	15138	2115	236	41105	41288	41751
cdpeo	2110	522	75	13621	2694	312	29051	28617	29315
emo	2652	1355	258	7781	2784	619	12524	12134	12552
gbm	3019	1074	168	12572	3284	503	19490	21330	20801
gfvo	1737	575	80	2828	1524	293	4058	4104	4003
koro	2006	591	80	5154	2272	382	7271	7181	7422
mamo	1984	511	61	3557	1488	234	5060	5059	4998
Overall	2086	693	108	8664	2309	368	16937	17102	17263

Table 5.3: Results of the evaluation of cache effectiveness. The mean number of reasoner calls required for a single application of the axiom weakening operator on randomly selected axioms of the ontology is given for different degrees of cache reuse. The cache is reused of one, ten, or one hundred successive operator applications.

We can see clearly from the results that the cache is indeed effective at lowering both the number of reasoner calls and execution time. The simple caching method alone provides a dramatic decrease in the number of reasoner calls, especially at higher group sizes. The algorithm that can additionally exploit the transitivity of the relation performs even better, also with a smaller number of weakening steps. The difference when looking at the execution time is significantly smaller. This can likely be attributed largely to internal caching performed by the reasoners. This would also explain the observed variation when it comes to the relative improvement between different reasoners. It can be concluded that the addition of the cache greatly benefits the computation of the axiom weakening operator, especially if it can be reused for many applications of the same weakening operator.

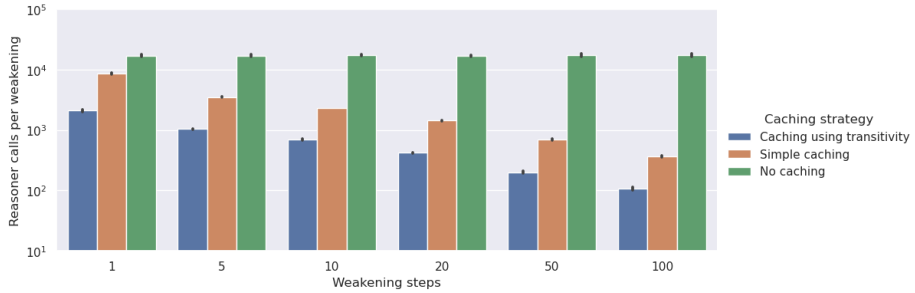


Figure 5.4: The mean number of reasoner calls needed for a single application of the axiom weakening operator, averaged over the tested ontologies.

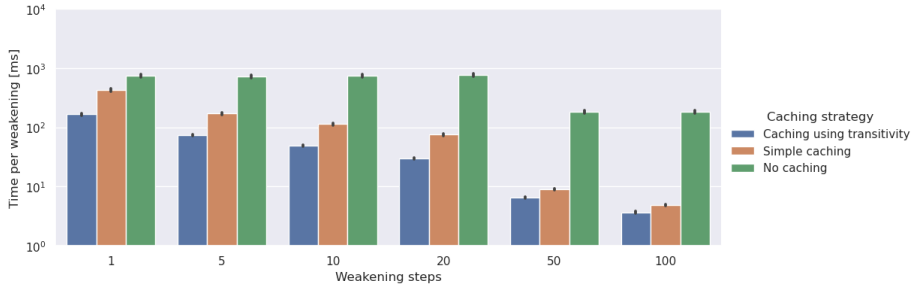


Figure 5.5: Average time required per application of the axiom weakening operator with different caching strategies. The results are averaged over the tested ontologies and the reasoners FaCT++, JFact, Openllet, and HermiT.

5.3 Reasoner Calls and Execution Time

The performance of the axiom weakening based repair algorithm shown in Algorithm 4.1 has also been evaluated. During each repair via axiom weakening, the required (real) time and the number of calls to the reasoner have been recorded. Additionally, the number of steps taken by the repair using axiom weakening has been observed. Also, as mentioned in Section 5.1, the repair program was put under a timeout to prevent cases where the reasoning becomes unreasonably slow. The generation of inconsistent ontologies used a similar timeout, and the same procedure was used for situations in which the reasoner ran out of memory. The timeouts of the weakening procedure are shown separately from those latter cases in the results. The frequency of these cases is indicated as the percentage of overall started runs that actually finished. The results of this evaluation are presented in Table 5.4 and Fig. 5.6.

As is visible from the results, the number of reasoner calls and the execution time can vary significantly. The execution times were generally reasonable when a run was able to complete within the time limit, with most of them completing within 2 minutes, even though the time limit was set to 5 minutes. A significant number of runs, however, were affected by the timeout or other errors. It has not been looked deeper into what causes these issues.

	Steps	Calls	Time [ms]	Failed
admin	6.3 [4.0, 9.2]	7621 [5601, 10118]	9638 [5433, 14909]	2% (2%)
ahso	2.1 [1.7, 2.5]	4648 [4228, 5131]	11469 [8122, 15336]	11% (26%)
cdao	2.1 [1.8, 2.5]	16137 [14576, 17740]	20767 [15204, 27269]	19% (48%)
cdpeo	1.5 [1.3, 1.7]	4476 [4250, 4716]	2050 [1942, 2169]	0% (0%)
covid19-ibo	1.3 [1.2, 1.5]	14822 [14321, 15320]	2208 [2126, 2296]	4% (13%)
ecp	1.3 [1.2, 1.5]	2020 [1916, 2133]	4453 [2738, 6939]	1% (14%)
emo	1.4 [1.3, 1.6]	18549 [17473, 19615]	6582 [4439, 9688]	1% (4%)
evi	5.0 [3.9, 6.4]	5955 [4680, 7425]	4719 [3408, 6349]	19% (64%)
falls	1.5 [1.3, 1.6]	1242 [1161, 1331]	874 [843, 909]	1% (5%)
fo	1.6 [1.4, 1.8]	1295 [1179, 1423]	1090 [1025, 1164]	38% (61%)
gbm	1.5 [1.3, 1.6]	7608 [7070, 8131]	2954 [2808, 3117]	0% (0%)
gfvo	1.8 [1.6, 2.0]	4167 [3864, 4501]	2404 [2258, 2569]	0% (0%)
koro	1.9 [1.7, 2.2]	6195 [5697, 6706]	2456 [2277, 2648]	0% (0%)
lico	2.6 [2.2, 2.9]	6638 [6105, 7186]	3709 [3400, 4077]	1% (22%)
mamo	2.5 [2.2, 2.9]	4667 [4228, 5128]	2215 [2046, 2403]	0% (0%)
mpio	2.2 [1.8, 2.6]	1908 [1720, 2133]	987 [913, 1078]	9% (30%)
pizza	2.0 [1.7, 2.3]	7550 [6723, 8419]	26767 [20554, 34014]	28% (55%)
provo	4.9 [3.7, 6.3]	6851 [5383, 8465]	8878 [5962, 12348]	4% (8%)
qudt	1.1 [1.0, 1.2]	7044 [6816, 7291]	6658 [4229, 9672]	9% (38%)
trans	3.2 [1.8, 5.1]	4384 [3213, 5894]	3684 [1610, 6513]	0% (5%)
triage	3.2 [2.4, 4.3]	5166 [4255, 6351]	8979 [6084, 13102]	28% (51%)
vio	2.5 [2.1, 3.0]	9476 [8628, 10327]	15199 [10496, 20803]	3% (7%)

Table 5.4: Results of the evaluation with respect to performance. The number of weakening iterations, reasoner calls, and total repair time are given as sample mean, with the 95% confidence interval in brackets. The frequency of failed runs is shown as percentage of repairs by weakening that were started but not completed. In parentheses, the percentage of total failed runs, including those with timeout during generation of the inconsistent ontology. Attempts that failed were not considered for the other values.

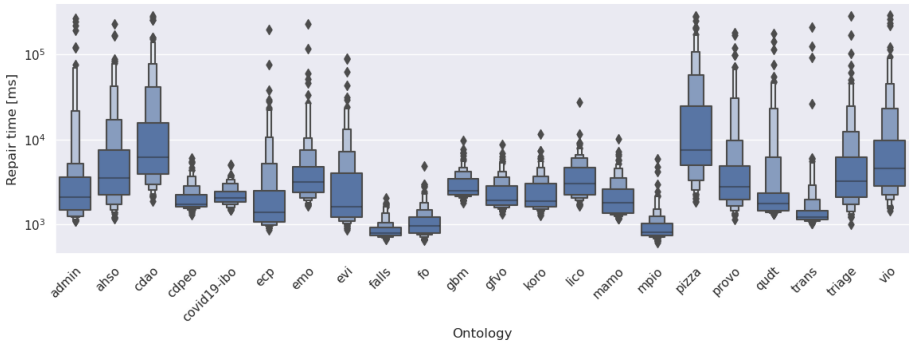


Figure 5.6: Distribution of (real) execution time required for repairing a single ontology using axiom weakening. The two darkest blue boxes represent (together) half of the samples, with the line in the middle indicating the median. Each lighter boxes each represent half as many samples of the boxes one step darker, and all remaining outliers are marked. Attempts that failed by a timeout or other error were not considered.

Chapter 6

Conclusion and Outlook

This thesis has proposed refinement operators and an axiom weakening operator covering all aspects of $\mathcal{SROIQ}$ and shown that for the repair of inconsistent ontologies, weakening can, in some cases, significantly outperform removal. The proposed approaches have been implemented both for performing the empirical evaluation and in a Protégé plugin that allows users to use the techniques described in this thesis for manually weakening axioms and automatically repairing ontologies. Some ideas implemented for and discussed in the thesis, like the repair approaches focusing on finding better repairs discussed in Section 4.1.4 or the relaxed RIA weakening mentioned in Section 3.2.1, have not yet been thoroughly studied and still have to be evaluated.

Better ways for users to interact with and guide the repair process may also be studied. It can also be seen that the repair algorithm likely needs better heuristics to steer the selection of bad and weakened axioms to achieve better repairs. This may also involve the use of domain-specific information and heuristics that could guide the repair process to choose repairs that also make sense from a modelling perspective. The termination of the proposed repair algorithm still needs to be studied, as has been done in [16]. Further additions to the refinement operators may also be studied in more detail, e.g., using non-simple roles in the upward and downward covers in certain contexts. Relaxing the allowed weakening for RIAs may also be considered, and to cover also extensions to regularity conditions such as those studied in [37]. Future work could further focus on finding robust measures for comparing the quality of repairs. Different applications for the axiom weakening and refinement operators, like those studied for concept combination in [54], are now also able to be explored using more expressive DLs.

Bibliography

[1] Dmitry Tsarkov et al. *FaCT++*. commit 650a50ce78a2f9c7fd60. 2017. URL: https://bitbucket.org/dtsarkov/factplusplus.

[2] Protégé developers et al. *Protege 5 Developer Documentation*. URL: https://protegewiki.stanford.edu/wiki/Protege5DevDocs (visited on 05/14/2023).

[3] Grigoris Antoniou and Frank Van Harmelen. *A semantic web primer*. MIT press, 2004.

[4] Franz Baader. "Least Common Subsumers and Most Specific Concepts in a Description Logic with Existential Restrictions and Terminological Cycles". In: *IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9-15, 2003*. Ed. by Georg Gottlob and Toby Walsh. Morgan Kaufmann, 2003, pp. 319–324.

[5] Franz Baader, Ian Horrocks, Carsten Lutz, and Uli Sattler. *An Introduction to Description Logic*. Cambridge University Press, 2017.

[6] Franz Baader, Francesco Kriegel, Adrian Nuradiansyah, and Rafael Peñaloza. "Making Repairs in Description Logics More Gentle". In: *Principles of Knowledge Representation and Reasoning: Proceedings of the Sixteenth International Conference, KR 2018, Tempe, Arizona, 30 October - 2 November 2018*. 2018, pp. 319–328.

[7] Franz Baader, Rafael Peñaloza, and Boontawee Suntisrivaraporn. "Pinpointing in the Description Logic $\mathcal{EL}^+$". In: *KI 2007: Advances in Artificial Intelligence, 30th Annual German Conference on AI, KI 2007, Osnabrück, Germany, September 10-13, 2007, Proceedings*. Ed. by Joachim Hertzberg, Michael Beetz, and Roman Englert. Vol. 4667. Lecture Notes in Computer Science. Springer, 2007, pp. 52–67.

[8] "OWL 2 Web Ontology Language Quick Reference Guide (Second Edition)". In: (2012). Ed. by Jie Bao, Elisa F. Kendall, Deborah L. McGuinness, Peter F. Patel-Schneider, Li Ding, and Ankesh Khandelwal. URL: https://www.w3.org/TR/2012/REC-owl2-quick-reference-20121211/.

[9] Dave Beckett and Brian McBride. "RDF/XML syntax specification (revised)". In: *W3C recommendation* 10.2.3 (2004).

[10] David Beckett and Tim Berners-Lee. "Turtle - Terse RDF Triple Language". In: *https://www.w3.org/TeamSubmission/turtle/* (2011).

[11] Roland Bernard, Oliver Kutz, and Nicolas Troquard. "Making Axiom Weakening Work in $\mathcal{SROIQ}$". unpublished. 2023.

[12] *BioPortal*. 2023. URL: https://bioportal.bioontology.org/ (visited on 06/03/2023).

[13] Roberto Confalonieri, Manfred Eppe, Marco Schorlemmer, Oliver Kutz, Rafael Peñaloza, and Enric Plaza. "Upward Refinement Operators for Conceptual Blending in the Description Logic $\mathcal{EL}^{++}$". In: *Annals of Mathematics and Artificial Intelligence* (2016). DOI: 10.1007/s10472-016-9524-8.

[14] Roberto Confalonieri, Manfred Eppe, Marco Schorlemmer, Oliver Kutz, Rafael Peñaloza, and Enric Plaza. "Upward Refinement Operators for Conceptual Blending in the Description Logic $\mathcal{EL}^{++}$". In: *Annals of Mathematics and Artificial Intelligence* 82.1-3 (2018). Ed. by Springer Netherlands, pp. 69–99. ISSN: 1012-2443.

[15] Roberto Confalonieri, Pietro Galliani, Oliver Kutz, Daniele Porello, Guendalina Righetti, and Nicolas Troquard. "Irresistible Refinement Operators for Expressive Description Logics". unpublished. 2022.

[16] Roberto Confalonieri, Pietro Galliani, Oliver Kutz, Daniele Porello, Guendalina Righetti, and Nicolas Troquard. "Towards Even More Irresistible Axiom Weakening". In: *Proceedings of the 33rd International Workshop on Description Logics (DL 2020)*. Ed. by Stefan Borgwardt and Thomas Meyer. Vol. 2663. CEUR-WS, 2020.

[17] Rémi Coulom. "Efficient selectivity and backup operators in Monte-Carlo tree search". In: *International conference on computers and games*. Springer. 2006, pp. 72–83.

[18] Nick Drummond, Alan Rector, Matthew Horridge, Chris Wroe, and Robert Stevens. *Pizza Ontology*. 2023. URL: https://protege.stanford.edu/ontologies/pizza/pizza.owl (visited on 05/27/2023).

[19] Jianfeng Du, Guilin Qi, and Xuefeng Fu. "A practical fine-grained approach to resolving incoherent OWL 2 DL terminologies". In: *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. 2014, pp. 919–928.

[20] W3C Owl Working Group et al. "OWL 2 Web Ontology Language Document Overview (Second Edition)". In: (2012). URL: https://www.w3.org/TR/2012/REC-owl2-overview-20121211/.

[21] Nicola Guarino and Pierdaniele Giaretta. "Ontologies and knowledge bases". In: *Towards very large knowledge bases* (1995), pp. 1–2.

[22] Peter Haase and Guilin Qi. "An analysis of approaches to resolving inconsistencies in DL-based ontologies". In: *Proc. of IWOD-07*. 2007, pp. 97–109.

[23] "OWL 2 Web Ontology Language Primer (Second Edition)". In: (2012). Ed. by Pascal Hitzler, Markus Krötzsch, Bijan Parsia, Peter F Patel-Schneider, and Sebastian Rudolph. URL: https://www.w3.org/TR/2012/REC-owl2-primer-20121211/.

[24] Matthew Horridge and Sean Bechhofer. "The OWL API: A Java API for OWL Ontologies". In: *Semantic Web* 2.1 (2011), pp. 11–21.

[25] Matthew Horridge, Bijan Parsia, and Ulrike Sattler. "Laconic and precise justifications in OWL". In: *The Semantic Web-ISWC 2008: 7th International Semantic Web Conference, ISWC 2008, Karlsruhe, Germany, October 26-30, 2008. Proceedings 7*. Springer. 2008, pp. 323–338.

[26] Matthew Horridge and Peter F Patel-Schneider. "OWL 2 web ontology language manchester syntax". In: *W3C Working Group Note* (2009).

[27] Ian Horrocks, Oliver Kutz, and Ulrike Sattler. "The Even More Irresistible $\mathcal{SROIQ}$". In: *Proceedings, Tenth International Conference on Principles of Knowledge Representation and Reasoning, Lake District of the United Kingdom, June 2-5, 2006*. Ed. by Patrick Doherty, John Mylopoulos, and Christopher A. Welty. AAAI Press, 2006, pp. 57–67.

[28] Ian Horrocks, Oliver Kutz, and Ulrike Sattler. "The Irresistible SRIQ." In: *OWLED*. 2005.

[29] Ian Horrocks and Ulrike Sattler. "A tableau decision procedure for". In: *Journal of automated reasoning* 39.3 (2007), pp. 249–276.

[30] Qiu Ji, Zhiqiang Gao, Zhisheng Huang, and Man Zhu. "Measuring effectiveness of ontology debugging systems". In: *Knowledge-Based Systems* 71 (2014), pp. 169–186.

[31] Ulrich Junker. "Preferred explanations and relaxations for over-constrained problems". In: *AAAI-2004*. 2004.

[32] Aditya Kalyanpur, Bijan Parsia, B Cuenca-Grau, and Evren Sirin. *Axiom pinpointing: Finding (precise) justifications for arbitrary entailments in SHOIN (owl-dl)*. Tech. rep. Technical report, UMIACS, 2005-66, 2006.

[33] Aditya Kalyanpur, Bijan Parsia, Matthew Horridge, Evren Sirin, et al. "Finding all justifications of OWL DL entailments". In: *ISWC/ASWC* 4825 (2007), pp. 267–280.

[34] Aditya Kalyanpur, Bijan Parsia, Evren Sirin, and Bernardo Cuenca Grau. "Repairing Unsatisfiable Concepts in OWL Ontologies". In: *The Semantic Web: Research and Applications, 3rd European Semantic Web Conference, ESWC 2006, Budva, Montenegro, June 11-14, 2006, Proceedings*. Ed. by York Sure and John Domingue. Vol. 4011. Lecture Notes in Computer Science. Springer, 2006, pp. 170–184.

[35] Aditya Kalyanpur, Bijan Parsia, Evren Sirin, and James Hendler. "Debugging unsatisfiable classes in OWL ontologies". In: *Web Semantics: Science, Services and Agents on the World Wide Web* 3.4 (2005), pp. 268–293.

[36] Yevgeny Kazakov. "$\mathcal{RIQ}$ and $\mathcal{SROIQ}$ Are Harder than $\mathcal{SHOIQ}$". In: *Principles of Knowledge Representation and Reasoning: Proceedings of the Eleventh International Conference, KR 2008, Sydney, Australia, September 16-19, 2008*. Ed. by Gerhard Brewka and Jérôme Lang. AAAI Press, 2008, pp. 274–284.

[37] Yevgeny Kazakov. "An Extension of Complex Role Inclusion Axioms in the Description Logic $\mathcal{SROIQ}$". In: *Automated Reasoning, 5th International Joint Conference, IJCAR 2010, Edinburgh, UK, July 16-19, 2010. Proceedings*. Ed. by Jürgen Giesl and Reiner Hähnle. Vol. 6173. Lecture Notes in Computer Science. Springer, 2010, pp. 472–486.

[38] Yevgeny Kazakov, Markus Krötzsch, and Frantisek Simancik. "ELK reasoner: architecture and evaluation." In: *ORE*. Citeseer. 2012.

[39] Yevgeny Kazakov, Markus Krötzsch, and František Simančík. "The Incredible ELK: From Polynomial Procedures to Efficient Reasoning with $\mathcal{EL}$ Ontologies". In: *Journal of automated reasoning* 53.1 (2014), pp. 1–61.

[40] C Maria Keet and Rolf Grütter. "Toward a systematic conflict resolution framework for ontologies". In: *Journal of Biomedical Semantics* 12.1 (2021), pp. 1–15.

[41] Markus Krötzsch. *Description logic rules*. Vol. 8. IOS Press, 2010.

[42] Joey Sik Chun Lam, Derek Sleeman, Jeff Z Pan, and Wamberto Vasconcelos. "A fine-grained approach to resolving unsatisfiable ontologies". In: *Journal on Data Semantics X*. Springer, 2008, pp. 62–95.

[43] Patrick Lambrix. "Completing and Debugging Ontologies: State of the Art and Challenges in Repairing Ontologies". In: *ACM Journal of Data and Information Quality* (2023).

[44] Domenico Lembo, Maurizio Lenzerini, Riccardo Rosati, Marco Ruzzi, and Domenico Fabio Savo. "Inconsistency-Tolerant Semantics for Description Logics". In: *Web Reasoning and Rule Systems - Fourth International Conference, RR 2010, Bressanone/Brixen, Italy, September 22-24, 2010. Proceedings*. Ed. by Pascal Hitzler and Thomas Lukasiewicz. Vol. 6333. Lecture Notes in Computer Science. Springer, 2010, pp. 103–117.

[45] Robert Malouf. "Maximal consistent subsets". In: *Computational Linguistics* 33.2 (2007), pp. 153–160.

[46] Joao Marques-Silva, Mikoláš Janota, and Anton Belov. "Minimal sets over monotone predicates in boolean formulae". In: *Computer Aided Verification: 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings 25*. Springer. 2013, pp. 592–607.

[47] Nicolas Matentzoglu and Ignazio Palmisano. "An Introduction to the OWL API". In: (2016). URL: http://syllabus.cs.manchester.ac.uk/pgt/2018/COMP62342/introduction-owl-api-msc.pdf.

[48] "OWL 2 Web Ontology Language Profiles (Second Edition)". In: (2012). Ed. by Boris Motik, Bernardo Cuenca Grau, Ian Horrocks, Zhe Wu, Achille Fokoue, and Carsten Lutz. URL: https://www.w3.org/TR/2012/REC-owl2-profiles-20121211/.

[49] Boris Motik and Peter Patel-Schneider. *OWL 2 web ontology language mapping to RDF graphs*. 2009.

[50] "OWL 2 Web Ontology Language. Structural Specification and Functional-Style Syntax (Second Edition)". In: (2012). Ed. by Boris Motik, Peter Patel-Schneider, and Bijan Parsia. URL: https://www.w3.org/TR/2012/REC-owl2-syntax-20121211/.

[51] Mark A Musen. "The Protégé project: A look back and a look forward". In: *AI matters* 1.4 (2015), pp. 4–12.

[52] Ignazio Palmisano, Simon Spero, Peter Ansell, Matthew Horridge, Stian Soiland-Reyes, Jim Balhoff, Nicolas Rouquette, Huseyin Kir, and Lewis John McGibbney et al. *OWLAPI*. commit b683b03cbe7587fbc8a1. 2023. URL: https://github.com/owlcs/owlapi.

[53] Raymond Reiter. "A theory of diagnosis from first principles". In: *Artificial intelligence* 32.1 (1987), pp. 57–95.

[54] Guendalina Righetti, Daniele Porello, Nicolas Troquard, Oliver Kutz, Maria Hedblom, and Pietro Galliani. "Asymmetric hybrids: Dialogues for computational concept combination". In: *Formal Ontology in Information Systems: Proceedings of the Twelfth International Conference (FOIS 2021)*. Vol. 344. IOS Press. 2022, p. 81.

[55] Patrick Rodler, Wolfgang Schmid, and Konstantin Schekotihin. "A generally applicable, highly scalable measurement computation and optimization approach to sequential model-based diagnosis". In: *arXiv preprint arXiv:1711.05508* (2017).

[56] Sebastian Rudolph. "Foundations of Description Logics". In: *Reasoning Web. Semantic Technologies for the Web of Data: 7th International Summer School 2011, Galway, Ireland, August 23-27, 2011, Tutorial Lectures 7* (2011), pp. 76–136.

[57] Konstantin Schekotihin, Patrick Rodler, Wolfgang Schmid, Philipp Fleiß, Dietmar Jannach, and Thomas Schmitz. *OntoDebug*. commit 1b1c5115537b679460cf. 2021. URL: `https://git-ainf.aau.at/interactive-KB-debugging/debugger`.

[58] Stefan Schlobach and Ronald Cornet. "Non-Standard Reasoning Services for the Debugging of Description Logic Terminologies". In: *IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9-15, 2003*. Ed. by Georg Gottlob and Toby Walsh. Morgan Kaufmann, 2003, pp. 355–362.

[59] Stefan Schlobach, Zhisheng Huang, Ronald Cornet, and Frank Van Harmelen. "Debugging incoherent terminologies". In: *Journal of Automated Reasoning* 39 (2007), pp. 317–349.

[60] Stefan Schulz. "The Role of Foundational Ontologies for Preventing Bad Ontology Design." In: *JOWO*. 2018.

[61] Kostyantyn Shchekotykhin, Dietmar Jannach, and Thomas Schmitz. "MergeXplain: Fast computation of multiple conflicts for diagnosis". In: *Twenty-Fourth International Joint Conference on Artificial Intelligence*. 2015.

[62] Rob Shearer and Ian Horrocks. "Exploiting partial information in taxonomy construction". In: *The Semantic Web-ISWC 2009: 8th International Semantic Web Conference, ISWC 2009, Chantilly, VA, USA, October 25-29, 2009. Proceedings 8*. Springer. 2009, pp. 569–584.

[63] Nicolas Troquard, Roberto Confalonieri, and Pietro Galliani. *OntologyUtils*. commit 331b422171e88fa63811. 2022. URL: `https://bitbucket.org/troquard/ontologyutils`.

[64] Nicolas Troquard, Roberto Confalonieri, Pietro Galliani, Rafael Peñaloza, Daniele Porello, and Oliver Kutz. "Repairing Ontologies via Axiom Weakening". In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*. Ed. by Sheila A. McIlraith and Kilian Q. Weinberger. AAAI Press, 2018, pp. 1981–1988.

[65] Dmitry Tsarkov and Ian Horrocks. "FaCT++ description logic reasoner: System description". In: *Automated Reasoning: Third International Joint Conference, IJCAR 2006, Seattle, WA, USA, August 17-20, 2006. Proceedings 3*. Springer. 2006, pp. 292–297.

[66] Jennifer Vendetti, Matthew Horridge, Igor Postanogov, and Tania Tudorache. *protege-plugin-examples*. commit d879601324d0c45d99e0. 2018. URL: `https://github.com/protegeproject/protege-plugin-examples`.

[67] Patricia L Whetzel, Natalya F Noy, Nigam H Shah, Paul R Alexander, Csongor Nyulas, Tania Tudorache, and Mark A Musen. "BioPortal: enhanced functionality via new Web services from the National Center for Biomedical Ontology to access and use ontologies in software applications". In: *Nucleic acids research* 39.suppl_2 (2011), W541–W545.

Appendix A

The $\mathcal{SROIQ}$ Description Logic

We give a brief description of the DL $\mathcal{SROIQ}$; for full details, see [27, 36, 5, 56].

A.1 $\mathcal{SROIQ}$ Syntax

The *vocabulary*[1] $N = N_I \cup N_C \cup N_R$ of a $\mathcal{SROIQ}$ ontology is made up of three finite disjoint sets:

- The set of *individual names* N_I is used to refer to single elements in the domain of discourse.
- The set of *concept names* N_C is used to refer to classes that elements of the domain may be a part of.
- The set of *role names* N_R is used to refer to binary relations that may hold between the elements of the domain.

A $\mathcal{SROIQ}$ ontology $\mathcal{O} = \mathcal{A} \cup \mathcal{T} \cup \mathcal{R}$ is the union of an *ABox* $\mathcal{A}$, a *TBox* $\mathcal{T}$, and a regular *RBox* $\mathcal{R}$. The elements of $\mathcal{O}$ are called axioms.

A.1.1 RBox

The RBox $\mathcal{R}$ describes the relationship between different roles in the ontology. It consists of two disjoint parts, a role hierarchy $\mathcal{R}_h$ and a set of role assertions $\mathcal{R}_a$.

Given the set of role names N_R, a *role* is either the *universal role* U or of the form r or r^- for some role name $r \in N_R$. r^- is called the *inverse role of* r. For convenience in the latter definitions, and to avoid roles like r^{--}, which are not valid in $\mathcal{SROIQ}$, we define a function inv such that $\mathrm{inv}(U) = U$, $\mathrm{inv}(r) = r^-$, and $\mathrm{inv}(r^-) = r$. We denote the set of all roles as $\mathcal{L}(N_R) = N_R \cup \{U\} \cup \{r^- \mid r \in N_R\}$.

A *role inclusion axiom* (RIA) is a statement of the form $R_1 \circ \cdots \circ R_n \sqsubseteq R$ where $R, R_1, \ldots R_n \in \mathcal{L}(N_R)$ are roles. For the case in which $n = 1$, we obtain a *simple role inclusion*, which has the form $R \sqsubseteq S$ where S and R are role names (the case where $n > 1$ is called a *complex role inclusion*). A finite set of RIAs is called a *role hierarchy*, denoted $\mathcal{R}_h$.

[1] There are no strict rules for how to write down different elements of the vocabulary. However, there is a convention of using PascalCase for concept names and camelCase for names referring to roles and individuals.

Roles can be partitioned into two disjoint sets, simple roles and non-simple roles. Intuitively, non-simple roles are those roles which are implied by the composition of two or more other roles. In order to preserve decidability, $\mathcal{SROIQ}$ requires that in some expressions only simple roles are used. We define the set of *non-simple roles* as the smallest set such that:

- the universal role U is non-simple,
- any role R that appears in a RIA of the form $S_1 \circ \cdots \circ S_n \sqsubseteq R$ where $n > 1$ is non-simple,
- any role R that appears in a simple RIA $S \sqsubseteq R$ where S is non-simple is itself non-simple, and
- if a role R is non-simple, then $\mathrm{inv}(R)$ is also non-simple.

All roles which are not non-simple are *simple roles*.

Example A.1. In the RBox $\mathcal{R} = \{r \circ s \sqsubseteq t, r \sqsubseteq s, t \sqsubseteq v^-\}$, the roles t and v^- are non-simple. t is non-simple because it appears as the super role in the complex RIA $r \circ s \sqsubseteq t$. v is non-simple because it appears on the right-hand side of the RIA $t \sqsubseteq v^-$, and t is non-simple. Since t and v^- are non-simple, t^- and v are also non-simple. The roles r and s on the other hand are simple. Even though, s appears as the super role in the RIA $r \sqsubseteq s$, since it is not a complex RIA and r is simple, it does not affect the simplicity of s.

There is an additional restriction that is placed upon the role hierarchy in a $SROIQ$ ontology. The role hierarchy in $\mathcal{SROIQ}$ must be regular. A role hierarchy $\mathcal{R}_h$ is *regular* if there exists a preorder $\preceq$ (that is, a transitive and reflexive relation) on the set of roles $\mathcal{L}(N_R)$, such that $S \prec R \iff \mathrm{inv}(S) \prec R$ for all roles R and S, and all RIA in $\mathcal{R}_h$ are $\preceq$-regular. A RIA is defined to be $\preceq$*-regular* if it is of one of the following forms:

- $\mathrm{inv}(R) \sqsubseteq R$,
- $R \circ R \sqsubseteq R$,
- $R \circ S_1 \circ \cdots \circ S_n \sqsubseteq R$,
- $S_1 \circ \cdots \circ S_n \circ R \sqsubseteq R$, or
- $S_1 \circ \cdots \circ S_n \sqsubseteq R$,

where $n > 1$ and R, S, $S_1, \cdots, S_n$ are roles such that $S \preceq R$, $S_i \preceq R$, and $R \not\preceq S_i$ for $i = 1, \ldots, n$. This condition on the role hierarchy prevents cyclic definitions with RIAs that include role chains. These types of cyclic definitions could otherwise lead to undecidability of the logic.

Example A.2. The RBox $\mathcal{R} = \{r \circ s \sqsubseteq t, t \sqsubseteq v, v \sqsubseteq s\}$ is not regular. Intuitively, there exists a cyclic dependency involving a complex RIA. That is, t depends on s via a complex RIA, v subsumes t, and s subsumes v. This means, $s \preceq t$, $t \preceq v$ and $v \preceq s$ must hold. By transitivity, $t \preceq s$ must also hold. However, because the first axiom is a complex RIA, $t \not\preceq s$ must hold, causing a contradiction.

To make axiom weakening simpler, this definition is slightly more general than necessary. The definition of regularity presented here is more permissive than the one in [27] in that it always allows simple roles on the left-hand side, similar to what has been described in [56]. However, it is more permissive than stated in [56] in that it allows for inverse roles on the right-hand side. This definition of regularity aligns more closely with the implementation of the OWL 2 DL [50] profile checker in the OWL API [24, 52].

The set of *role assertions* $\mathcal{R}_a$ is a finite set of statements with the form $\mathrm{disjoint}(S_1, S_2)$ (*disjointness*) where S_1, and S_2 are simple roles in $\mathcal{R}_h$. In [27] the authors define additionally the role assertions $\mathrm{Sym}(R)$ (*symmetry*), $\mathrm{Asy}(S)$ (*asymmetry*), $\mathrm{Tra}(R)$ (*transitivity*), $\mathrm{Ref}(R)$ (*reflexivity*), and $\mathrm{Irr}(R)$ (*irreflexivity*). These additional assertions can, however, be written using the alternative sets of axioms $\{\mathrm{inv}(R) \sqsubseteq R\}$, $\{\mathrm{disjoint}(R, \mathrm{inv}(R))\}$, $\{R \circ R \sqsubseteq R\}$, $\{r' \sqsubseteq R, \top \sqsubseteq \exists r'.\mathit{Self}\}$, and $\{\top \sqsubseteq \neg\exists R.\mathit{Self}\}$ respectively. Note that the asymmetry assertion requires a simple role, and that r' in the case of reflexivity must be a fresh role name not otherwise used in the ontology[2].

A.1.2 TBox

The TBox $\mathcal{T}$ describes the relationship between different concepts. In $\mathcal{SROIQ}$, the set of *concept expressions* (or simply *concepts*) given an RBox $\mathcal{R}$ is inductively defined as the smallest set such that:

- $\top$ and $\bot$ are concepts, respectively called *top concept* and *bottom concept*,
- all concept names $C \in N_C$ are concept, called *atomic concepts*,
- sets of individual names $\{a\}$ with $a \in N_I$ are concepts, called *nominal concepts*,
- if C and D are concepts, the $\neg C$ (*negation*), $C \sqcup D$ (*union*), and $C \sqcap D$ (*intersection*) are also concepts,
- if C is a concept and $R \in \mathcal{L}(N_R)$ a (possibly non-simple) role, then $\exists R.C$ (*existential quantification*) and $\forall r.C$ (*universal quantification*) are also concepts, and
- if C is a concept, $S \in \mathcal{L}(N_R)$ a simple role and $n \in \mathbb{N}_0$ a non-negative number, then $\exists S.\mathit{Self}$ (*Self restriction*), $\leq\ n\ S.C$ (*at-most restriction*), and $\geq\ n\ S.C$ (*at-least restriction*) are concepts, the last two may together be called *qualified number restrictions*.

Given two concepts C and D, a *concept inclusion axiom* (GCI) is a statement of the form $C \sqsubseteq D$. The TBox $\mathcal{T}$ is a finite set of such GCIs.

A.1.3 ABox

The ABox $\mathcal{A}$ contains statements about single individuals called individual assertions. An *individual assertion* has one of the following forms:

- $C(a)$ (*concept assertion*) for some concept C and individual name $a \in N_I$,

[2] This is necessary to allow the use of non-simple roles in a reflexivity assertion. Multiple assertions can share the same role name r'.

- $R(a, b)$ (*role assertion*) or $\neg R(a, b)$ (*negative role assertion*) for some role $R \in \mathcal{L}(N_R)$ and individual names $a, b \in N_I$, or
- $a = b$ (*equality*) or $a \neq b$ (*inequality*) for some individual names $a \in N_I$.

An ABox $\mathcal{A}$ is a finite set of such individual assertions. In $\mathcal{SROIQ}$ due to the inclusion of nominal concepts, all ABox axioms can be rewritten into TBox axioms.

A.2 $\mathcal{SROIQ}$ Semantics

A.2.1 Interpretations

The semantics of $\mathcal{SROIQ}$, similar to other description logics, are defined in a model-theoretic way. Therefore, a central notion is that of the interpretation. And *interpretation* $I = \langle \Delta^I, \cdot^I \rangle$ consists of a set Δ^I called the *domain* of I, and an *interpretation function* $\cdot^I$. The interpretation function maps the vocabulary elements as follows:

- Each individual name $a \in N_I$ is mapped to an element $a^I \in \Delta^I$ in the domain.
- Each concept name $C \in N_C$ is mapped to a subset $C^I \subseteq \Delta^I$ of the domain.
- Each role name $r \in N_R$ is mapped to a relation $r^I \subseteq \Delta^I \times \Delta^I$ over the domain.

An interpretation maps the universal role U to $U^I = \Delta^I \times \Delta^I$. We extend the interpretation function to operate over inverse roles such that $(r^-)^I$ contains exactly those elements $\langle \delta_1, \delta_2 \rangle$ for which $\langle \delta_2, \delta_1 \rangle$ is contained in r^I, that is $(r^-)^I = \{\langle \delta_1, \delta_2 \rangle \mid \langle \delta_2, \delta_1 \rangle \in r^I\}$. Further, we define the extension of the interpretation function to complex concepts inductively as follows:

- The top concept is true for every individual in the domain, therefore $\top^I = \Delta^I$.
- The bottom concept is true for no individual, hence $\bot^I = \emptyset$ where $\emptyset$ represents the empty set.
- Nominal concepts contain exactly the specified individuals, that is $\{a\}^I = \{a^I\}$.
- $\neg C$ yields the complement of the extension of C, thus $(\neg C)^I = \Delta^I \setminus C^I$.
- $C \sqcup D$ denotes all individuals that are in either the extension of C or in that of D, hence $(C \sqcup D)^I = C^I \cup D^I$.
- $C \sqcap D$ on the other hand, denotes all elements of the domain that are in the extension of both C and D, which can be expressed as $(C \sqcap D)^I = C^I \cap D^I$.
- $\exists R.C$ holds for all individuals that are connected by some element in the extension of R to an individual in the extension of C, formally $(\exists R.C)^I = \{\delta_1 \in \Delta^I \mid \exists \delta_2 \in \Delta^I \,.\, \langle \delta_1, \delta_2 \rangle \in R^I \wedge \delta_2 \in C^I\}$.
- $\forall R.C$ refers to all domain elements for which all elements in the extension of R connect them to elements in the extension of C, that is $(\forall R.C)^I = \{\delta_1 \in \Delta^I \mid \forall \delta_2 \in \Delta^I \,.\, \langle \delta_1, \delta_2 \rangle \in R^I \rightarrow \delta_2 \in C^I\}$.
- $\exists R.\mathit{Self}$ indicates all individuals that the extension of R connects to themselves, hence we let $(\exists R.\mathit{Self})^I = \{\delta \in \Delta^I \mid \langle \delta, \delta \rangle \in R^I\}$.

- $\leq\ n\ R.C$ represents those individuals that have at most n other individuals they are R-related to in the extension of the concept C, that is $(\leq\ n\ R.C)^I = \{\delta_1 \in \Delta^I \mid \big|\{\delta_2 \in \Delta^I \mid \langle \delta_1, \delta_2 \rangle \in R^I\ \wedge\ \delta_2 \in C^I\}\big| \leq n\ \}$, where $|X|$ denotes the cardinality of a set X.

- $\geq n\ R.C$ corollary to the case above indicates the elements of the domain that have at least N such R-related elements,
  $(\geq n\ R.C)^I = \{\delta_1 \in \Delta^I \mid \big|\{\delta_2 \in \Delta^I \mid \langle \delta_1, \delta_2 \rangle \in R^I\ \wedge\ \delta_2 \in C^I\}\big| \geq n\ \}$.

Example A.3. Given the interpretation $I = \langle \Delta^I, \cdot^I \rangle$ where $\Delta^I = \{1, 2, 3, 4, 5, 6, 7\}$ and $P^I = \{2, 3, 5, 7\}$, $E^I = \{2, 4, 6\}$, $o^I = 1$, and $s^I = \{\langle 1, 2\rangle, \langle 2, 3\rangle, \langle 3, 4\rangle, \langle 4, 5\rangle, \langle 5, 6\rangle, \langle 6, 7\rangle\}$, the extended interpretation function evaluates the following example as shown.

$$\top^I = \{1, 2, 3, 4, 5, 6, 7\} \qquad (\neg P)^I = \{1, 4, 6\}$$
$$(P \cup E)^I = \{2, 3, 4, 5, 6, 7\} \qquad (P \cap E)^I = \{2\}$$
$$(\exists s.P)^I = \{1, 2, 4, 6\} \qquad (\exists s.E \cup \exists s^-.E)^I = \{1, 3, 5, 7\}$$
$$(\exists s^-.\{o\})^I = \{2\} \qquad (\exists s.\{o\})^I = \emptyset$$

A.2.2 Satisfaction of Axioms

The main purpose of this extended interpretation function is to determine the satisfaction of axioms. We define now when an axiom α is true, or holds, in a specific interpretation I. If this is the case, the interpretation I satisfies α, written $I \models \alpha$. We also say that I is a model of α.

- A RIA $S_1 \circ \cdots \circ S_n \sqsubseteq R$ holds in I if and only if for each sequence $\delta_1, \ldots, \delta_{n+1} \in \Delta^I$ for which $\langle \delta_i, \delta_{i+1} \rangle \in S_i^I$ for all $i = 1, \cdots, n$, also $\langle \delta_1, \delta_n \rangle \in R\rangle$ is satisfied. Equivalently, we can write $I \models S_1 \circ \cdots \circ S_n \sqsubseteq R \iff S_1^I \circ \cdots \circ S_n^I \subseteq R^I$ where $\circ$ denotes the composition of relations.

- A role disjointness axiom $\mathrm{disjoint}(S, R)$ hold if and only if the extensions of R and S are disjoint, formally $I \models \mathrm{disjoint}(S, R) \iff S^I \cap R^I = \emptyset$.

- A GCI $C \sqsubseteq D$ is true if and only if the extension of C is fully contained in the extension of D, hence $I \models C \sqsubseteq D \iff C^I \subseteq D^I$.

- A concept assertion $C(a)$ holds if and only if the individual that a is mapped to by $\cdot^I$ is in the extension of the concept C, therefore $I \models C(a) \iff a^I \in C^I$.

- A role assertion $R(a, b)$ holds if and only if the individuals denoted by the names a and b are connected in the extension of R, thus $I \models R(a, b) \iff \langle a^I, b^I \rangle \in R^I$.

- A negative role assertion $\neg R(a, b)$ is true exactly then when the corresponding role assertion $R(a, b)$ is false. Equivalently, $I \models \neg R(a, b) \iff \langle a^I, b^I \rangle \notin R^I$.

- An equality assertion $a = b$ holds if and only if the individuals identified by a and b are the same element of the domain, formally written $I \models a = b \iff a^I = b^I$.

- Dual to the above, $a \neq b$ holds if and only if the names a and b denote different elements, accordingly $I \models a \neq b \iff a^I \neq b^I$.

We say a set of axioms holds in an interpretation I if and only if every axiom of the set hold in I. Accordingly, I satisfies a ontology $\mathcal{O}$, written $I \models \mathcal{O}$, if and only if I satisfies every axiom $\alpha \in \mathcal{O}$ of the ontology, i.e., $I \models \mathcal{O} \iff \forall \alpha \in \mathcal{O} : I \models \alpha$. If I satisfies $\mathcal{O}$, we say I is a model of $\mathcal{O}$.

Example A.4. Given the interpretation $I = \langle \Delta^I, \cdot^I \rangle$ where $\Delta^I = \{1, 2, 3, 4, 5, 6, 7\}$ and $P^I = \{2, 3, 5, 7\}$, $E^I = \{2, 4, 6\}$, $o^I = 1$, $t^I = 2$, and $s^I = \{\langle 1, 2\rangle, \langle 2, 3\rangle, \langle 3, 4\rangle, \langle 4, 5\rangle, \langle 5, 6\rangle, \langle 6, 7\rangle\}$, the following axioms are satisfied by I.

$$
\begin{array}{ll}
P \sqsubseteq \top & P \sqcap \neg(\exists s^-.\{o\}) \sqsubseteq \neg E \\
P \sqcup E \sqsubseteq \neg\{o\} & \mathrm{disjoint}(s, s^-) \\
s(o, t) & o \neq t
\end{array}
$$

The following axioms on the other hand are not satisfied by I.

$$
\begin{array}{lll}
P \sqsubseteq \neg E & \top \sqsubseteq P \sqcup E & \top \sqsubseteq \exists s.\top \\
s \circ s \sqsubseteq s & s \sqsubseteq s^- & \top \sqsubseteq \geq 1\ s.E \\
\neg s(o, t) & o = t & s(t, o)
\end{array}
$$

A.2.3 Reasoning tasks

In general, logic-based knowledge representation is useful for the ability to perform reasoning tasks on ontologies. Several reasoning tasks can be performed by a reasoner in DLs. In this section, we will have a look at three basic reasoning tasks, and how they can be reduced to each other. While there exist other reasoning tasks, this section will focus on *ontology consistency*, *axiom entailment*, and *concept satisfiability*.

Ontology consistency

An ontology $\mathcal{O}$ is consistent if and only if there exists a model $I \models \mathcal{O}$ for $\mathcal{O}$. Otherwise, the ontology is called inconsistent. As discussed in Section 2.3, an inconsistent ontology can be a sign of modelling errors. An inconsistent ontology entailed every statement, and as such all information extracted from it is useless. Therefore, an inconsistent ontology is generally undesirable. Furthermore, both the task of deciding concept satisfiability and axiom entailment can be reduced to deciding ontology consistency.

Axiom entailment

An axiom α is entailed by $\mathcal{O}$ if every model $I \models \mathcal{O}$ of the ontology also satisfies the axiom $I \models \alpha$. We also write this as $\mathcal{O} \models \alpha$ and say that α is a consequence of $\mathcal{O}$. Deciding axiom entailment is an important task in order to derive new information from the collected knowledge. If α is entailed by the empty ontology, α is said to be a *tautology*. Further, the *set of consequences* of $\mathcal{O}$ if the set of all axioms which are entailed by $\mathcal{O}$, we write $\mathrm{Con}(\mathcal{O}) = \{\alpha \mid \mathcal{O} \models \alpha\}$. It is clear that the set of consequences will always be infinite since there is an infinite number of tautologies.

The problem of axiom entailment can be reduced to determining the satisfiability of a modified ontology. This is achieved by using an axiom β that imposes the opposite restriction to α, to be more precise, for all interpretations $I \models \alpha \iff I \not\models \beta$. If α is entailed by $\mathcal{O}$, it must hold in every model of $\mathcal{O}$, hence β must not hold in any model.

It follows that the extended ontology $\mathcal{O} \cup \{ \beta \}$ has no new model, and is therefore unsatisfiable. We can consequently solve the axiom entailment problem by testing for the satisfiability of a modified ontology if we can find such an opposing axiom for α. For some cases in $\mathcal{SROIQ}$ finding such an opposite is obvious, for others the desired behaviour must be emulated with a set of axioms. Appendix A.2.3 shows the correspondence for every type of $\mathcal{SROIQ}$ axiom.

α	B
$S_1 \circ \cdots \circ S_n \sqsubseteq R$	$S_1(a_1, a_2), \ldots, S_n(a_n, a_{n+1})$, and $\neg R(a_1, a_{n+1})$
$\text{disjoint}(S, R)$	$S(a, b)$ and $R(a, b)$
$C \sqsubseteq D$	$(C \sqcap \neg D)(a)$
$C(a)$	$\neg C(a)$
$R(a, b)$	$\neg R(a, b)$
$\neg R(a, b)$	$R(a, b)$
$a = b$	$a \neq b$
$a \neq b$	$a = b$

Table A.1: The axioms in B together have the "opposite" meaning of those in α. a, a_i, and b are assumed to not appear in $\mathcal{O}$. This means, checking the entailment of α with respect to $\mathcal{O}$, is equivalent to checking the unsatisfiability of $\mathcal{O} \cup B$.

Concept satisfiability

A concept C is satisfiable with respect to $\mathcal{O}$ if and only if there exists a model of the ontology $I \models \mathcal{O}$ such that the extension of C is not empty, i.e., $C^I \neq \emptyset$. A concept which is not satisfiable is called unsatisfiable. Clearly, some concepts are unsatisfiable with respect to every ontology, for example, $\bot$ or $A \sqcap \neg A$. However, similar to an unsatisfiable ontology, an unsatisfiable atomic concept may be an indication of a modelling mistake.

Like axiom entailment, concept satisfiability can be reduced to ontology satisfiability. If a concept is unsatisfiable, every model $I \models \mathcal{O}$ maps the concept to the empty set, that is $C^I = \emptyset$. Since the other direction is trivial, we can rewrite this as $C^I \subseteq \emptyset$. It follows that since $\bot^I = \emptyset$ every such model satisfies $I \models C \sqsubseteq \bot$, meaning $\mathcal{O} \models C \sqsubseteq \bot$. We conclude that we can test for the unsatisfiability of a concept C by checking for entailment of $C \sqsubseteq \bot$.

Appendix B

Axiom Weakening in OWL 2 DL

Since OWL 2 DL is reducible to $\mathcal{SROIQ}$, it would be sufficient to perform this normalization and then apply the weakening as described to the resulting $\mathcal{SROIQ}$ ontology. This transformation is unproblematic in some contexts, for example, if the result is only going to be used for automatic reasoning tasks. However, if the output must be further manipulated by a user of the system, the added noise introduced by the normalization may cause confusion and hinder understanding. Further, weakening OWL 2 DL ontologies directly can be seen as a heuristic, giving an indication as to which weakening might make sense from a modelling perspective.

Example B.1. OWL has an axiom $\mathrm{DisjointClasses}(C_1, \ldots, C_n)$[1] that allows specifying that a set of classes all are pairwise disjoint. $C_i \sqcap C_j \sqsubseteq \bot$ for all $1 \leq i < j \leq n$. One reasonable approach to weakening the OWL axiom is to replace any of the classes C_i with a more specific class $C_i' \in \rho_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(C_i)$. In contrast, after normalization, there will be $n-1$ occurrences of C_i. It is unlikely, increasingly so with growing n, that all such occurrences will be weakened to the same concept. After weakening the normalized ontology, it is thus in general not possible to reconstruct the disjointness axiom.

It should be noted that working directly with OWL 2 axioms will make repairs less gentle. For some axiom types, it is not obvious how they could reasonably be weakened to another single axiom. For these kinds of axioms, removal is the only available weakening.

Example B.2. The OWL axiom $\mathrm{EquivalentClasses}(C_1, \ldots, C_n)$ can not easily be weakened. One option for weakening is removing one of the arguments. The axiom would be normalized to a set of $\mathrm{SubClassOf}$ axioms, for which both the subclass and superclasses can be modified. It is evident that this is more gentle than completely removing arguments.

For OWL 2 DL we must follow the same restrictions, when it comes to regularity and simplicity of roles, as for $\mathcal{SROIQ}$. The same definitions for the upward and downward covers, $\mathrm{UpCover}_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}$ and $\mathrm{DownCover}_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}$, as in Definition 3.1 are used.

Definition B.1. We define the *abstract refinement operator* $\zeta_{\uparrow,\downarrow}$ for OWL 2 DL class expressions as follows.

[1] In the rest of this and the following section, the OWL 2 functional syntax (cf. [50]) will be used.

$$
\begin{aligned}
\zeta_{\uparrow,\downarrow}(A) &= \uparrow(A) \qquad \text{for } A \in \mathrm{N}_c \cup \mathcal{L}(N_R) \cup \{\top, \bot\} \\
\zeta_{\uparrow,\downarrow}(\mathrm{ObjectComplementOf}(C)) &= \uparrow(\mathrm{ObjectComplementOf}(C)) \\
&\quad \cup \{\mathrm{ObjectComplementOf}(C') \mid C' \in \zeta_{\downarrow,\uparrow}(C)\} \\
\zeta_{\uparrow,\downarrow}(\mathrm{ObjectIntersectionOf}(C_1, \ldots, C_n)) &= \uparrow(\mathrm{ObjectIntersectionOf}(C_1, \ldots, C_n)) \\
&\quad \cup \bigcup_{i=1}^{n} \{\mathrm{ObjectIntersectionOf}(C_1, \ldots, C_i', \ldots C_n) \mid C_i' \in \zeta_{\uparrow,\downarrow}(C_i)\} \\
\zeta_{\uparrow,\downarrow}(\mathrm{ObjectUnionOf}(C_1, \ldots, C_n)) &= \uparrow(\mathrm{ObjectUnionOf}(C_1, \ldots, C_n)) \\
&\quad \cup \bigcup_{i=1}^{n} \{\mathrm{ObjectUnionOf}(C_1, \ldots, C_i', \ldots C_n) \mid C_i' \in \zeta_{\uparrow,\downarrow}(C_i)\} \\
\zeta_{\uparrow,\downarrow}(\mathrm{ObjectAllValuesFrom}(R, C)) &= \uparrow(\mathrm{ObjectAllValuesFrom}(R, C)) \\
&\quad \cup \{\mathrm{ObjectAllValuesFrom}(R', C) \mid R' \in \zeta_{\downarrow,\uparrow}(R)\} \\
&\quad \cup \{\mathrm{ObjectAllValuesFrom}(R, C') \mid C' \in \zeta_{\uparrow,\downarrow}(C)\} \\
\zeta_{\uparrow,\downarrow}(\mathrm{ObjectSomeValuesFrom}(R, C)) &= \uparrow(\mathrm{ObjectSomeValuesFrom}(R, C)) \\
&\quad \cup \{\mathrm{ObjectSomeValuesFrom}(R', C) \mid R' \in \zeta_{\uparrow,\downarrow}(R)\} \\
&\quad \cup \{\mathrm{ObjectSomeValuesFrom}(R, C') \mid C' \in \zeta_{\uparrow,\downarrow}(C)\} \\
\zeta_{\uparrow,\downarrow}(\mathrm{ObjectHasSelf}(R)) &= \uparrow(\mathrm{ObjectHasSelf}(R)) \\
&\quad \cup \{\mathrm{ObjectHasSelf}(R') \mid R' \in \zeta_{\uparrow,\downarrow}(R)\} \\
\zeta_{\uparrow,\downarrow}(\mathrm{ObjectMaxCardinality}(n, R, C)) &= \uparrow(\mathrm{ObjectMaxCardinality}(n, R, C)) \\
&\quad \cup \{\mathrm{ObjectMaxCardinality}(n', R, C) \mid n' \in \uparrow(n)\} \\
&\quad \cup \{\mathrm{ObjectMaxCardinality}(n, R', C) \mid R' \in \zeta_{\downarrow,\uparrow}(R)\} \\
&\quad \cup \{\mathrm{ObjectMaxCardinality}(n, R, C') \mid C' \in \zeta_{\downarrow,\uparrow}(C)\} \\
\zeta_{\uparrow,\downarrow}(\mathrm{ObjectMinCardinality}(n, R, C)) &= \uparrow(\mathrm{ObjectMinCardinality}(n, R, C)) \\
&\quad \cup \{\mathrm{ObjectMinCardinality}(n', R, C) \mid n' \in \downarrow(n)\} \\
&\quad \cup \{\mathrm{ObjectMinCardinality}(n, R', C) \mid R' \in \zeta_{\uparrow,\downarrow}(R)\} \\
&\quad \cup \{\mathrm{ObjectMinCardinality}(n, R, C') \mid C' \in \zeta_{\uparrow,\downarrow}(C)\} \\
\zeta_{\uparrow,\downarrow}(\mathrm{ObjectOneOf}(a_1, \ldots, a_n)) &= \uparrow(\mathrm{ObjectOneOf}(a_1, \ldots, a_n))
\end{aligned}
$$

OWL 2 concepts:

$$
\begin{aligned}
\zeta_{\uparrow,\downarrow}(\mathrm{ObjectExactCardinality}(n, R, C)) &= \uparrow(\mathrm{ObjectExactCardinality}(n, R, C)) \\
&\quad \cup \{\phi_1 \sqcap \phi_2 \mid \phi_1 \in \zeta_{\uparrow,\downarrow}(\mathrm{ObjectMaxCardinality}(n, R, C)) \\
&\qquad \wedge \phi_2 \in \zeta_{\uparrow,\downarrow}(\mathrm{ObjectMinCardinality}(n, R, C))\} \\
\zeta_{\uparrow,\downarrow}(\mathrm{ObjectHasValue}(R, a)) &= \uparrow(\mathrm{ObjectHasValue}(R, a)) \\
&\quad \cup \{\mathrm{ObjectHasValue}(R', a) \mid R' \in \zeta_{\uparrow,\downarrow}(R)\} \\
&\quad \cup \{\mathrm{ObjectSomeValuesFrom}(R, A) \mid A \in \zeta_{\uparrow,\downarrow}(\{a\})\}
\end{aligned}
$$

Using this abstract refinement operator, we build two concrete refinement operators, a *generalization operator* $\gamma_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}} = \zeta_{\mathrm{UpCover}_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}},\mathrm{DownCover}_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}}$ and a *specialization operator* $\rho_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}} = \zeta_{\mathrm{DownCover}_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}},\mathrm{UpCover}_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}}$.

Using these generalization and specialization operators, we then define the axiom weakening operator $g_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}$ for OWL 2 DL axioms as follows. Note that the axiom $\bot \sqsubseteq \top$ is not actually an OWL 2 axiom, but stands in for some tautological axiom that is true in every possible interpretation.

Definition B.2. The *axiom weakening operator* for OWL 2 DL axioms is defined as follows.

$$
\begin{aligned}
g_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(\mathrm{SubClassOf}(C, D)) &= \{\mathrm{SubClassOf}(C', D) \mid C' \in \rho_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(C)\} \\
&\quad \cup \{\mathrm{SubClassOf}(C, D') \mid D' \in \gamma_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(D)\} \\
g_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(\mathrm{ClassAssertion}(C, a)) &= \{\mathrm{ClassAssertion}(C', a) \mid C' \in \gamma_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(C)\}
\end{aligned}
$$

$$
\begin{aligned}
g_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(\text{ObjectPropertyAssertion}(R,a,b)) =& \{\text{ObjectPropertyAssertion}(R',a,b) \mid R' \in \gamma_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(R)\} \\
& \cup \{\text{ObjectPropertyAssertion}(R,a,b), \bot \sqsubseteq \top\} \\
g_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(\text{NegativeObjectPropertyAssertion}(R,a,b)) =& \{\text{NegativeObjectPropertyAssertion}(R',a,b) \mid R' \in \rho_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(R)\} \\
& \cup \{\text{NegativeObjectPropertyAssertion}(R,a,b), \bot \sqsubseteq \top\} \\
g_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(\text{SameIndividual}(a_1,\ldots,a_n)) =& \bigcup_{i=1}^{n} \{\text{SameIndividual}(\{a_1,\ldots,a_n\} \setminus \{a_i\})\} \\
& \cup \{\text{SameIndividual}(a_1,\ldots,a_n)\} \\
g_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(\text{DifferentIndividuals}(a_1,\ldots,a_n)) =& \bigcup_{i=1}^{n} \{\text{DifferentIndividuals}(\{a_1,\ldots,a_n\} \setminus \{a_i\})\} \\
& \cup \{\text{DifferentIndividuals}(a_1,\ldots,a_n)\} \\
g_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(\text{DisjointObjectProperties}(r_1,\ldots,r_n)) =& \bigcup_{i=1}^{n} \{\text{DisjointObjectProperties}(r_1,\ldots,r_i',\ldots,r_n) \mid r_i' \in \rho_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(r_i)\} \\
g_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(\text{SubObjectPropertyOf}(S,R)) =& \{\text{SubObjectPropertyOf}(S',R) \mid S' \in \rho_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(S)\} \\
& \cup \{\text{SubObjectPropertyOf}(S,R') \mid R' \in \gamma_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(R) \wedge R \text{ is simple}\} \\
& \cup \{\bot \sqsubseteq \top\} \\
g_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(\text{SubObjectPropertyOf}(& \\
\text{ObjectPropertyChain}(s_1,\ldots,s_n),R)) =& \{\text{SubObjectPropertyOf}(\\
& \quad \text{ObjectPropertyChain}(s_1,\ldots,s_i',\ldots,s_n),R) \mid s_i' \in \rho_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(s_i)\} \\
& \cup \{\text{SubObjectPropertyOf}(\\
& \quad \text{ObjectPropertyChain}(s_1,\ldots,s_n),R)\}
\end{aligned}
$$

OWL 2 axioms:

$$
\begin{aligned}
g_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(\text{EquivalentClasses}(C_1,\ldots,C_n)) =& \bigcup_{i=1}^{n} \{\text{EquivalentClasses}(\{C_1,\ldots,C_n\} \setminus \{C_i\})\} \\
& \cup \{\text{EquivalentClasses}(C_1,\ldots,C_n)\} \\
g_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(\text{DisjointClasses}(C_1,\ldots,C_n)) =& \bigcup_{i=1}^{n} \{\text{DisjointClasses}(C_1,\ldots,C_i',\ldots,C_n) \mid C_i' \in \rho_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(C_i)\} \\
g_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(\text{DisjointUnion}(D,C_1,\ldots,C_n)) =& \{\text{DisjointUnion}(D,C_1,\ldots,C_n), \bot \sqsubseteq \top\} \\
g_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(\text{EquivalentObjectProperties}(r_1,\ldots,r_n)) =& \bigcup_{i=1}^{n} \{\text{EquivalentObjectProperties}(\{r_1,\ldots,r_n\} \setminus \{r_i\})\} \\
& \cup \{\text{EquivalentObjectProperties}(r_1,\ldots,r_n)\} \\
g_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(\text{InverseObjectProperties}(S,R)) =& \{\text{InverseObjectProperties}(S,R), \bot \sqsubseteq \top\} \\
g_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(\text{FunctionalObjectProperty}(R)) =& \{\text{FunctionalObjectProperty}(R), \bot \sqsubseteq \top\} \\
g_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(\text{InverseFunctionalObjectProperty}(R)) =& \{\text{InverseFunctionalObjectProperty}(R), \bot \sqsubseteq \top\} \\
g_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(\text{SymmetricObjectProperty}(R)) =& \{\text{SymmetricObjectProperty}(R), \bot \sqsubseteq \top\} \\
g_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(\text{AsymmetricObjectProperty}(R)) =& \{\text{AsymmetricObjectProperty}(R), \bot \sqsubseteq \top\} \\
g_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(\text{TransitiveObjectProperty}(R)) =& \{\text{TransitiveObjectProperty}(R), \bot \sqsubseteq \top\} \\
g_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(\text{ReflexiveObjectProperty}(R)) =& \{\text{ReflexiveObjectProperty}(R), \bot \sqsubseteq \top\} \\
g_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(\text{IrreflexiveObjectProperty}(R)) =& \{\text{IrreflexiveObjectProperty}(R), \bot \sqsubseteq \top\} \\
g_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(\text{ObjectPropertyDomain}(R,C)) =& \{\text{ObjectPropertyDomain}(R,C') \mid C' \in \gamma_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(C)\} \\
g_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(\text{ObjectPropertyRange}(R,C)) =& \{\text{ObjectPropertyRange}(R,C') \mid C' \in \gamma_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}(C)\}
\end{aligned}
$$

Appendix C

Mapping from OWL 2 DL to $\mathcal{SROIQ}$

Since OWL 2 DL has some features that are not present in $\mathcal{SROIQ}$ it is necessary to perform some normalization in order to apply the weakening as described in Definition 3.3. We define to this end a translation π that maps OWL 2 DL class expressions, object property expressions, and axioms respectively to $\mathcal{SROIQ}$ concepts, roles, and axioms. Note that a single OWL 2 axiom is mapped to a set of $\mathcal{SROIQ}$ axioms that together have the equivalent meaning. This is necessary because not all OWL 2 axioms can be translated into a single $\mathcal{SROIQ}$ axiom. The translation does not cover annotations or axioms relating to data properties, as those do not map directly to $\mathcal{SROIQ}$, and were removed in the preprocessing step for the evaluation. Further, declaration axioms are not mentioned here, but they are assumed to be used for generating the vocabulary N_C, N_R, and N_I.

Definition C.1. The *translation* π from OWL 2 DL object property expressions to $\mathcal{SROIQ}$ roles is give by

$$\begin{aligned} \pi(r) &= r \quad \text{for } r \in N_R \cup \{U, E\} \ , \\ \pi(\text{InverseObjectProperty}(R)) &= \text{inv}(\pi(R)) \ . \end{aligned}$$

The translation from OWL 2 DL class expressions to $\mathcal{SROIQ}$ concepts is given by

$$\begin{aligned} \pi(A) &= A \quad \text{for } A \in N_C \cup \{\top, \bot\} \ , \\ \pi(\text{ObjectComplementOf}(C)) &= \neg\pi(C) \ , \\ \pi(\text{ObjectIntersectionOf}(C_1, \ldots, C_n)) &= ((C_1 \sqcap C_2) \cdots \sqcap C_n) \ , \\ \pi(\text{ObjectUnionOf}(C_1, \ldots, C_n)) &= ((C_1 \sqcup C_2) \cdots \sqcup C_n) \ , \\ \pi(\text{ObjectAllValuesFrom}(R, C)) &= \forall\pi(R).\pi(C) \ , \\ \pi(\text{ObjectSomeValuesFrom}(R, C)) &= \exists\pi(R).\pi(C) \ , \\ \pi(\text{ObjectHasSelf}(R)) &= \exists\pi(R).\mathit{Self} \ , \\ \pi(\text{ObjectMaxCardinality}(n, R, C)) &= \leq n\, \pi(R).\pi(C) \ , \\ \pi(\text{ObjectMinCardinality}(n, R, C)) &= \geq n\, \pi(R).\pi(C) \ , \\ \pi(\text{ObjectOneOf}(a_1, \ldots, a_n)) &= ((\{a_1\} \sqcup \{a_2\}) \cdots \sqcup \{a_n\}) \ , \\ \pi(\text{ObjectExactCardinality}(n, R, C)) &= (\geq n\, \pi(R).\pi(C)) \sqcup (\leq n\, \pi(R).\pi(C)) \ , \\ \pi(\text{ObjectHasValue}(R, a)) &= \exists\pi(R).\{a\} \ . \end{aligned}$$

The translation from OWL 2 DL axioms to $\mathcal{SROIQ}$ axioms is given by

$$
\begin{aligned}
\pi(\text{SubClassOf}(C, D)) &= \{\pi(C) \sqsubseteq \pi(D)\} \ , \\
\pi(\text{ClassAssertion}(C, a)) &= \{\pi(C)(a)\} \ , \\
\pi(\text{ObjectPropertyAssertion}(R, a, b)) &= \{\pi(R)(a, b)\} \ , \\
\pi(\text{NegativeObjectPropertyAssertion}(R, a, b)) &= \{\neg\pi(R)(a, b)\} \ , \\
\pi(\text{SameIndividual}(a_1, \ldots, a_n)) &= \{a_i = a_j \mid 1 \leq i < j \leq n\} \ , \\
\pi(\text{DifferentIndividuals}(a_1, \ldots, a_n)) &= \{a_i \neq a_j \mid 1 \leq i < j \leq n\} \ , \\
\pi(\text{DisjointObjectProperties}(R_1, \ldots, R_n)) &= \{\text{disjoint}(\pi(R_i), \pi(R_j)) \mid 1 \leq i < j \leq n\} \ , \\
\pi(\text{SubObjectPropertyOf}(S, R)) &= \{\pi(S) \sqsubseteq \pi(R)\} \ , \\
\pi(\text{SubObjectPropertyOf}(& \\
\text{ObjectPropertyChain}(S_1, \ldots, S_n), R)) &= \{\pi(S_1) \circ \cdots \circ \pi(S_n) \sqsubseteq \pi(R)\} \ , \\
\pi(\text{EquivalentClasses}(C_1, \ldots, C_n)) &= \{\pi(C_i) \sqsubseteq \pi(C_j) \mid 1 \leq i \leq n \text{ and } 1 \leq j \leq n \text{ and } i \neq j\} \ , \\
\pi(\text{DisjointClasses}(C_1, \ldots, C_n)) &= \{\pi(C_i) \sqcap \pi(C_j) \sqsubseteq \bot \mid 1 \leq i < j \leq n\} \ , \\
\pi(\text{DisjointUnion}(D, C_1, \ldots, C_n)) &= \{\pi(C_i) \sqcap \pi(C_j) \sqsubseteq \bot \mid 1 \leq i < j \leq n\} \\
&\quad \cup \{D \sqsubseteq ((C_1 \sqcup C_2) \cdots \sqcup C_n), ((C_1 \sqcup C_2) \cdots \sqcup C_n) \sqsubseteq D\} \ , \\
\pi(\text{EquivalentObjectProperties}(R_1, \ldots, R_n)) &= \{\pi(R_i) \sqsubseteq \pi(R_j) \mid 1 \leq i \leq n \text{ and } 1 \leq j \leq n \text{ and } i \neq j\} \ , \\
\pi(\text{InverseObjectProperties}(S, R)) &= \{\text{inv}(\pi(S)) \sqsubseteq \pi(R), \pi(R) \sqsubseteq \text{inv}(\pi(S))\} \ , \\
\pi(\text{FunctionalObjectProperty}(R)) &= \{\top \sqsubseteq \ \leq 1 \ \pi(R).\top\} \ , \\
\pi(\text{InverseFunctionalObjectProperty}(R)) &= \{\top \sqsubseteq \ \leq 1 \ \text{inv}(\pi(R)).\top\} \ , \\
\pi(\text{SymmetricObjectProperty}(R)) &= \{\text{inv}(\pi(R)) \sqsubseteq \pi(R)\} \ , \\
\pi(\text{AsymmetricObjectProperty}(R)) &= \{\text{disjoint}(\text{inv}(\pi(R)), \pi(R))\} \ , \\
\pi(\text{TransitiveObjectProperty}(R)) &= \{\pi(R) \circ \pi(R) \sqsubseteq \pi(R)\} \ , \\
\pi(\text{ReflexiveObjectProperty}(R)) &= \{\top \sqsubseteq \exists\pi(R).\mathit{Self}\} \ , \\
\pi(\text{IrreflexiveObjectProperty}(R)) &= \{\top \sqsubseteq \neg\exists\pi(R).\mathit{Self}\} \ , \\
\pi(\text{ObjectPropertyDomain}(R, C)) &= \{\exists\pi(R).\top \sqsubseteq \pi(C)\} \ , \\
\pi(\text{ObjectPropertyRange}(R, C)) &= \{\exists \, \text{inv}(\pi(R)).\top \sqsubseteq \pi(C)\} \ .
\end{aligned}
$$

Appendix D

User Guide: OntologyUtils

This section will explain how to use the prototype that has been implemented for the evaluation. There are two main ways in which someone may want to use the implementation. First, as a Java library for axiom weakening. This use case will only be discussed briefly. Second, the prototype may be used to apply the axiom weakening based repair to OWL ontology files using the implemented applications.

D.1 Building the Software

The prototype is implemented in Java and will require at least Java version 17. The Maven tool has been used for dependency management. Both Maven and a Java 17 JDK must be installed for building this project. To download and build the project, the following code can be executed. After building, the packaged JAR files will be located in `target/`. There will be two different packages, one containing only the code in the project named `ontologyutils-X.X.X.jar`, and one containing also all required dependencies named `shaded-ontologyutils-X.X.X.jar`.

```
git clone https://github.com/rolandbernard/ontologyutils
cp ontologyutils
mvn clean compile package
```

If you would rather use the pre-packaged jar files, those are also available in the GitHub repository at `https://github.com/rolandbernard/ontologyutils/releases`. Note that the releases may be out of date with the master branch of the repository.

D.2 Use as a Library

Since the project is implemented as a Maven project, it is possible to use it as a dependency for another Maven project. To add the library as a dependency, if you have built the project yourself and run `mvn install`, it is sufficient to add the following to your `pom.xml` file.

```
<dependency>
  <groupId>ontologyutils</groupId>
  <artifactId>ontologyutils</artifactId>
  <version>0.1.0</version>
</dependency>
```

There is also the possibility to let Maven handle the installation from the repository by adding the following configuration to the `pom.xml` file. This will add the GitHub Maven repository and download the dependency from the ones available at `https://github.com/rolandbernard?tab=packages&repo_name=ontologyutils`.

```
<repositories>
    <repository>
        <id>github</id>
        <url>https://maven.pkg.github.com/rolandbernard/*</url>
    </repository>
</repositories>
```

Javadoc comments are included for most classes and methods, so it should be rather easy to understand the structure of the project.

D.3 Using the Applications

To simply use the applications that were also used for the evaluation in this thesis, use the JAR file produced during building. The packaged output at `target/shaded-ontologyutils-X.X.X.jar` will include all the necessary dependencies. From there it is enough to run the JAR file using a command if the form `java -cp <jar-file > www.ontologyutils.apps.<app-class> [argument ...]`, where `<jar-file>` should be replaced with the path to the JAR package and `<app-class>` should be replaced with one of the following.

- `BenchCache` Is a utility application used for the evaluation of the cache effectiveness. It is not intended to be used by an end user.
- `Benchmark` Is another utility for benchmarking the performance of the axiom weakening implementation.
- `CheckConsistency` Is an application to test the consistency of an ontology, possibly using multiple different reasoner implementations.
- `ClassifyOntology` Can be used to determine whether an ontology fits into a certain OWL 2 profile. Further, it also shows how expressive the description logic features used are.
- `CleanupOntology` This application was used for the evaluation to apply normalization and remove axioms violating the OWL 2 DL profile.
- `EvaluateRepairs` Can be used to compute the size of the inferred class hierarchy and the IIC between different ontologies.
- `MakeInconsistent` Can be used to make a consistent ontology inconsistent by adding stronger versions of the axioms until they cause inconsistency.
- `RepairMcs` Can be used to repair an ontology using a randomly sampled maximal consistent subset.
- `RepairRemoval` Enables the repair of an ontology using the repair algorithm using removal.
- `RepairWeakening` Enables the repair of an ontology using the repair algorithm using axiom weakening.

- `ShowOntology` Is a simple application that prints all the axioms in the ontology.

To get specific information about how to use an application, simply use the `--help` argument. For example, the following is the help output for the `RepairWeakening` application.

```
Usage: www.ontologyutils.apps.RepairWeakening [options] <file>
Options:
  -h --help                print this help information and quit
  -o --output=<file>       the file to write the result to
  -n --normalize           normalize the ontology before repair
  --normalize-nnf          normalize the ontology to NNF before repair
  -R --no-repair           no not perform repair
  -v --verbose             print more information
  -V --extra-verbose       print even more information
  --limit=<integer>        number of repairs to generate
  --no-limit=<integer>     only stop once all repairs have been generated
  --reasoner={fact++|hermit|jfact|openllet}
                           the reasoner to use
  --coherence              make the ontology coherent
  --fast                   use fast methods for selection
  --ref-ontology={any|intersect|intersect-of-some|largest|random|random-of-
      some}
                           method for reference ontology selection
  --bad-axiom={largest-mcs|least-mcs|most-mus|one-mcs|one-mus|random|some-
      mcs|some-mus}
                           method for bad axiom selection
  --strict-nnf             accept and produce only NNF axioms
  --strict-alc             accept and produce only ALC axioms
  --strict-sroiq           accept and produce only SROIQ axioms
  --strict-simple-roles    use only simple roles in upward and downward
      covers
  --uncached               do not use any caches for the covers
  --basic-cache            use only a basic cache
  --strict-owl2            does not produce intersection and union with a
      single operand
  --simple-ria-weakening   do not use the more advanced RIA weakening
  --no-role-refinement     do not refine roles in any context
  --enhance-ref            keep the reference ontology as static axioms in
      the output
  --preset={bernard2023|confalonieri2020|troquard2018}
                           configuration approximating description in
                               papers
  <file>                   the file containing the original ontology
```

To repair the ontology in the file `inconsistent.owl` using the axiom weakening based repair algorithm, and write the repaired ontology to `consistent.owl`, for example, using the configuration used for the evaluation in this thesis, one would execute the following command.

```
java -cp target/shaded-ontologyutils-0.1.0.jar www.ontologyutils.apps.
    RepairWeakening inconsistent.owl -o consistent.owl
```

Appendix E

User Guide: Protégé Plugin

This section will cover how to install and use the Protégé plugin developed to allow using the techniques presented in the thesis.

E.1 Installation

The Protégé plugin can either be compiled and packaged from source or is available at `https://github.com/rolandbernard/protege-weakening/releases`. If you desire to compile the plugin from source, you will need a Java 11 JDK and the Maven tool. If you have both of these, you can execute the following command to package the plugin, and thereafter, find the bundled JAR file in `target/protege-weakening-X.X.X.jar`.

```
git clone https://github.com/rolandbernard/protege-weakening
cp protege-weakening
mvn clean compile package
```

The JAR file of the plugin must be placed into the plugin folder of Protégé. The location of this folder will vary depending on the platform and configuration of Protégé. In a Linux-based environment, for example, it will most likely be located in `$HOME/.Protege/plugins/`. After copying the JAR file of the plugin into this folder, the plugin will automatically be loaded and initialized the next time Protégé is started. Note that the plugin will require at least Java 11 and Protégé version 5.6.1. The plugin may work for some older Protégé versions, but this has not been tested.

E.2 Using the Menu

The easiest way to use the plugin is by using the added menu items in the "Tools" section of the top menu. There are three main menu items. The "Normalize to SROIQ" entry will execute the normalization of the ontology that has also been used for the evaluation in the thesis. This will transform all OWL 2 axioms into axioms that more directly correspond to $\mathcal{SROIQ}$. Note that doing this is not required for starting the repair algorithms, since the axiom weakening operator implemented in the plugin has been extended to handle all OWL 2 axioms.

The next two menu items are for applying two different automatic repair algorithms. For each of them, there is the option to select between making the ontology only consistent or making it also coherent. Note that when selecting coherence as

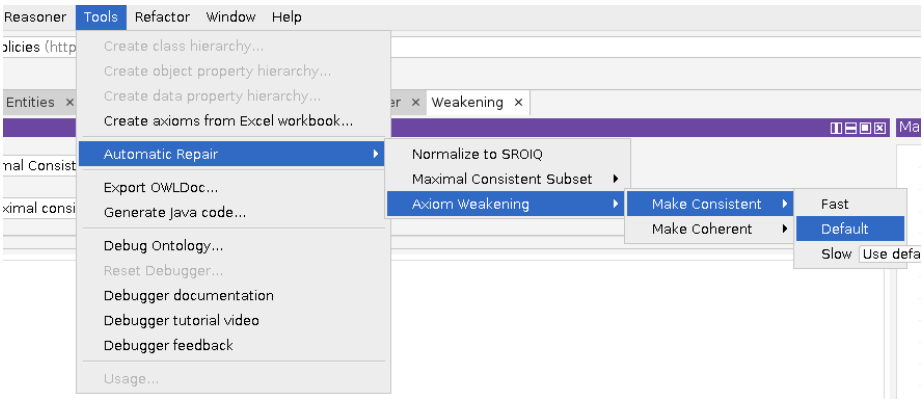


Figure E.1: The menu items added by the plugin in Protégé.

the goal, the repairs will likely be slower, also because computing coherence is more expensive, at least using the way it is implemented for the plugin. The first repair method, used with the item "Maximal Consistent Subset", will select some randomly sampled maximal consistent subset. The second repair method, "Axiom Weakening", on the other hand, uses the axiom weakening based repair algorithm. Note that for each of these algorithms, there is also a choice between "Fast", "Default", or "Slow". These are some presets that make different choices of how to configure the repair algorithms. As their name suggests, they mainly consider options that have an impact on the performance of the algorithm. To have greater control over these parameters, use the automatic repair view.

E.3 Using the Automatic Repair View

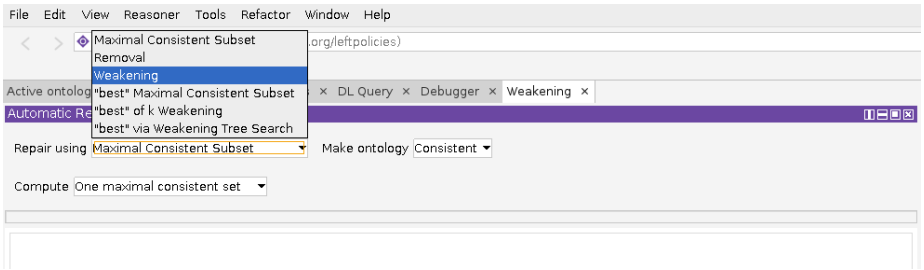


Figure E.2: The automatic repair view added to Protégé. Showing the selection of possible algorithms.

The automatic repair view can be accessed in "Window" > "Views" > "Ontology views" > "Automatic Repair". It allows the user to select between a number of different repair algorithms. The one using axioms weakening and the one using a maximal consistent subset are the same as the ones accessible through the menu. The repair by removal is similar to the axiom weakening based repair, but the axioms are always removed instead of being weakened. The remaining three algorithms are all trying to optimize the IIC of the resulting repairs. For "'best' Maximal Consistent Subset" and

"best' of k Weakening" this is done by sampling repairs using, respectively, maximal consistent subsets or axiom weakening, and then selecting from those the one that has the largest inferred call hierarchy. The algorithm used when selecting "'best' via Weakening Tree Search", on the other hand, uses a Monte-Carlo tree search based approach to find the ontology with the largest inferred class hierarchy. Note that these last three repairs are significantly slower, both because they need to compute multiple repairs, and because they have to compute the inferred class hierarchy for each of them.

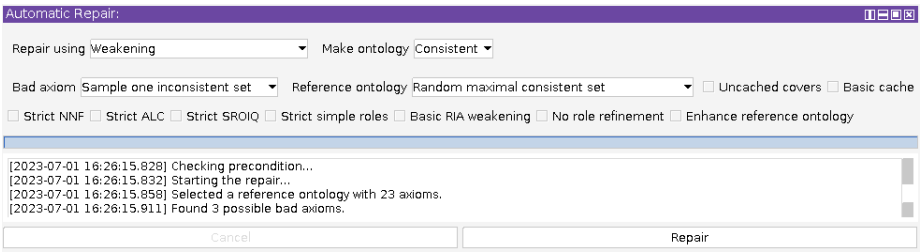


Figure E.3: The automatic repair view added to Protégé. Showing the different parameters that can be configured for the axiom weakening based repair method.

For each of these repair algorithms, a number of different configuration options are made available to the user. Figure E.3 shows the parameters that may be configured for the axiom weakening based repair algorithm. This includes different approaches for selecting reference ontologies and bad axioms. There are some options that allow using stricter checks, such as only allowing negation normal form, or only $\mathcal{ALC}$ axioms and concepts. Further, some parameters simplify the refinement of roles and one that chooses to not modify the axioms of the reference ontology.

To start the repair, use the "Repair" button at the bottom of the view. Some information about the running repair will be printed on the text output. While the repair is running, it can be interrupted by using the "Cancel" button that will become enabled whenever a repair is running. If a repair is cancelled, no change will be made to the ontology in Protégé. As soon as the repair completes, on the other hand, the changes will be applied to the active ontology.

E.4 Using the Manual Weakening View

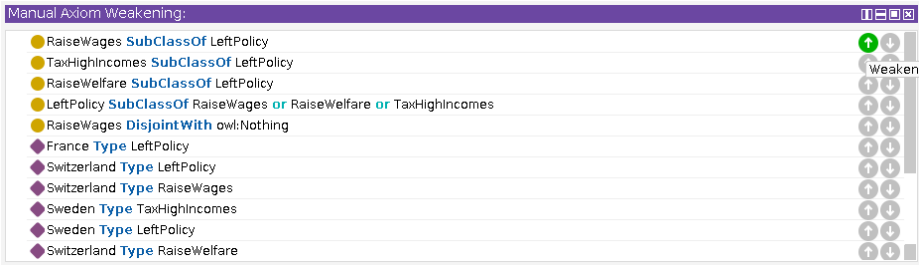


Figure E.4: The manual axiom weakening list added to Protégé. Use the buttons next to the axioms to weaken or strengthen the axioms.

The last view implemented by the plugin, and accessible under “Window” > “Views” > “Ontology views” > “Manual Axiom Weakening”, allows for manually selecting which axioms to refine. The view shows a list of all axioms in the currently active ontology. The axioms are sorted by how frequently they appear in randomly sampled minimal inconsistent subsets. The axioms appearing the most often are shown at the top of the list. This means, that the axioms that would be chosen for weakening by the automatic repair algorithms are the ones appearing first in the list. Next to each axiom are two buttons, one with an upwards and one with a downwards arrow. These can be used to, respectively, compute the axiom weakenings or axiom strengthening operator for that axiom.

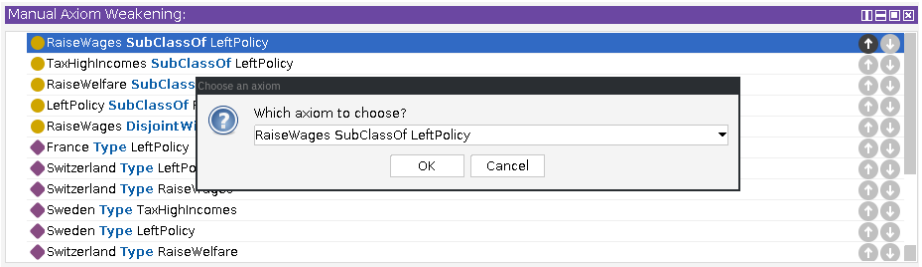


Figure E.5: The selection of which refined axiom to use for replacing the original axiom, for the manual axiom weakening list added to Protégé.

Pressing one of these buttons will bring up a selection dialogue showing the results of the axiom weakening or strengthening operator. The user may then select from the presented choices one axiom and click “Ok”. The original axiom will be removed from the ontology and replaced with the selected axiom. If the “Cancel” button is used on the other hand, no change will be made to the ontology.

Appendix F

Additional Evaluation Results

This appendix contains some more results produced by the evaluation. Most of these are not particularly more interesting than the ones already shown in Chapter 5 and have therefore, and for space reasons, not been included in the main part of the thesis.

IIC of	Weakening adding to reference ontology			Weakening		MCS
w.r.t.	Weakening	MCS	Removal	MCS	Removal	Removal
admin	0.64 [0.59, 0.69]	0.51 [0.45, 0.57]	0.67 [0.62, 0.72]	0.39 [0.33, 0.45]	0.56 [0.51, 0.60]	0.66 [0.61, 0.71]
ahso	0.63 [0.58, 0.68]	0.64 [0.59, 0.69]	0.70 [0.66, 0.74]	0.52 [0.47, 0.56]	0.57 [0.52, 0.62]	0.52 [0.48, 0.56]
cdao	0.55 [0.49, 0.61]	0.55 [0.48, 0.61]	0.58 [0.52, 0.65]	0.51 [0.44, 0.57]	0.54 [0.49, 0.61]	0.51 [0.45, 0.57]
cdpeo	0.67 [0.62, 0.72]	0.39 [0.33, 0.45]	0.66 [0.62, 0.71]	0.25 [0.20, 0.30]	0.50 [0.47, 0.53]	0.74 [0.69, 0.79]
covid19-ibo	0.54 [0.50, 0.58]	0.72 [0.68, 0.76]	0.77 [0.73, 0.81]	0.62 [0.58, 0.66]	0.71 [0.67, 0.75]	0.53 [0.50, 0.56]
ecp	0.68 [0.63, 0.73]	0.50 [0.44, 0.56]	0.86 [0.82, 0.90]	0.36 [0.30, 0.42]	0.75 [0.70, 0.80]	0.74 [0.70, 0.79]
emo	0.61 [0.57, 0.66]	0.73 [0.68, 0.77]	0.77 [0.72, 0.81]	0.61 [0.56, 0.65]	0.68 [0.63, 0.72]	0.53 [0.49, 0.57]
evi	0.63 [0.58, 0.68]	0.71 [0.66, 0.75]	0.68 [0.63, 0.74]	0.58 [0.53, 0.63]	0.51 [0.46, 0.55]	0.42 [0.38, 0.47]
falls	0.61 [0.56, 0.66]	0.60 [0.54, 0.66]	0.79 [0.73, 0.84]	0.50 [0.44, 0.57]	0.75 [0.70, 0.81]	0.64 [0.59, 0.69]
fo	0.44 [0.39, 0.50]	0.67 [0.62, 0.73]	0.48 [0.43, 0.54]	0.68 [0.63, 0.73]	0.53 [0.48, 0.58]	0.34 [0.29, 0.39]
gbm	0.61 [0.55, 0.66]	0.68 [0.63, 0.73]	0.69 [0.64, 0.74]	0.54 [0.49, 0.59]	0.59 [0.54, 0.64]	0.51 [0.46, 0.56]
gfvo	0.66 [0.61, 0.70]	0.71 [0.67, 0.75]	0.72 [0.68, 0.77]	0.54 [0.49, 0.59]	0.53 [0.47, 0.58]	0.46 [0.42, 0.51]
koro	0.64 [0.58, 0.69]	0.53 [0.47, 0.59]	0.68 [0.63, 0.73]	0.38 [0.32, 0.43]	0.53 [0.48, 0.58]	0.64 [0.59, 0.69]
lico	0.61 [0.56, 0.67]	0.63 [0.58, 0.68]	0.69 [0.64, 0.75]	0.51 [0.46, 0.57]	0.53 [0.48, 0.59]	0.48 [0.43, 0.53]
mamo	0.59 [0.53, 0.64]	0.72 [0.66, 0.77]	0.68 [0.63, 0.74]	0.64 [0.58, 0.69]	0.55 [0.49, 0.61]	0.43 [0.38, 0.49]
mpio	0.52 [0.46, 0.57]	0.70 [0.65, 0.74]	0.67 [0.62, 0.72]	0.70 [0.66, 0.75]	0.69 [0.65, 0.74]	0.37 [0.33, 0.41]
pizza	0.54 [0.48, 0.59]	0.63 [0.58, 0.69]	0.61 [0.55, 0.66]	0.57 [0.51, 0.63]	0.56 [0.51, 0.62]	0.48 [0.42, 0.53]
provo	0.62 [0.57, 0.67]	0.64 [0.60, 0.69]	0.64 [0.59, 0.68]	0.55 [0.51, 0.60]	0.51 [0.47, 0.56]	0.43 [0.39, 0.47]
qudt	0.59 [0.55, 0.64]	0.56 [0.49, 0.62]	0.94 [0.91, 0.97]	0.47 [0.40, 0.53]	0.96 [0.93, 0.98]	0.75 [0.71, 0.79]
trans	0.60 [0.54, 0.65]	0.51 [0.45, 0.57]	0.64 [0.59, 0.69]	0.41 [0.35, 0.47]	0.56 [0.51, 0.61]	0.65 [0.60, 0.71]
triage	0.63 [0.58, 0.69]	0.63 [0.57, 0.68]	0.65 [0.59, 0.71]	0.52 [0.47, 0.57]	0.51 [0.46, 0.56]	0.46 [0.41, 0.51]
vio	0.58 [0.51, 0.64]	0.57 [0.51, 0.63]	0.57 [0.51, 0.63]	0.50 [0.44, 0.57]	0.48 [0.41, 0.54]	0.44 [0.38, 0.50]
Overall	0.60 [0.58, 0.61]	0.61 [0.60, 0.63]	0.69 [0.67, 0.70]	0.52 [0.50, 0.53]	0.60 [0.58, 0.61]	0.53 [0.52, 0.55]

Table F.1: Results of the evaluation. IIC is given as mean and 95% confidence interval in brackets.

$\lvert\mathrm{Inf}(\mathcal{O})\rvert$ of	Weakening[a]	Weakening	MCS	Removal
admin	290 [265, 317]	255 [237, 274]	318 [287, 352]	218 [211, 226]
ahso	757 [730, 784]	704 [680, 728]	717 [691, 743]	712 [685, 738]
cdao	2843 [2494, 3202]	2592 [2317, 2889]	2841 [2529, 3178]	2637 [2364, 2937]
cdpeo	179 [176, 183]	171 [169, 173]	193 [187, 199]	171 [169, 173]
covid19-ibo	6832 [6499, 7155]	6696 [6364, 7001]	6517 [6185, 6839]	6362 [6028, 6673]
ecp	345 [320, 370]	271 [256, 286]	412 [384, 441]	260 [247, 274]
emo	6625 [6328, 6925]	6318 [6027, 6622]	6115 [5830, 6420]	6055 [5763, 6353]
evi	785 [764, 804]	766 [745, 786]	736 [714, 756]	771 [750, 792]
falls	248 [232, 265]	234 [219, 251]	247 [232, 263]	225 [211, 240]
fo	979 [959, 997]	980 [957, 1001]	924 [900, 947]	964 [936, 989]
gbm	6062 [5850, 6269]	5796 [5577, 6007]	5827 [5621, 6020]	5687 [5463, 5910]
gfvo	8766 [8613, 8907]	8202 [8018, 8373]	7940 [7733, 8133]	8126 [7925, 8319]
koro	1243 [1156, 1343]	1015 [986, 1046]	1311 [1214, 1419]	991 [967, 1017]
lico	6532 [6284, 6766]	6150 [5870, 6417]	6251 [5996, 6504]	6166 [5881, 6433]
mamo	9071 [8647, 9462]	8945 [8558, 9304]	8208 [7812, 8591]	8422 [7974, 8858]
mpio	479 [458, 500]	483 [462, 503]	413 [395, 430]	465 [446, 484]
pizza	6694 [6268, 7102]	6437 [6071, 6814]	6404 [6046, 6759]	6077 [5695, 6461]
provo	759 [742, 774]	737 [720, 753]	714 [695, 733]	738 [721, 754]
qudt	560 [508, 617]	539 [491, 594]	613 [549, 686]	524 [478, 578]
trans	408 [389, 428]	379 [369, 390]	441 [413, 471]	363 [355, 371]
triage	775 [742, 808]	727 [695, 759]	743 [712, 773]	750 [717, 784]
vio	2264 [2009, 2530]	2074 [1814, 2337]	2017 [1788, 2255]	2087 [1826, 2362]

Table F.2: Results of the evaluation. The cardinality of the inferred class hierarchy is given as mean and 95% confidence interval in brackets. [a]Weakening based repair that keeps the reference ontology as static axioms that are not weakened or replaced.

IIC^+ of	Weakening adding to reference ontology			Weakening		MCS
w.r.t.	Weakening	MCS	Removal	MCS	Removal	Removal
admin	0.52 [0.47, 0.58]	0.59 [0.52, 0.64]	0.57 [0.51, 0.62]	0.55 [0.48, 0.61]	0.56 [0.51, 0.60]	0.46 [0.40, 0.52]
ahso	0.66 [0.61, 0.71]	0.72 [0.67, 0.77]	0.78 [0.73, 0.83]	0.54 [0.48, 0.59]	0.63 [0.57, 0.68]	0.52 [0.47, 0.57]
cdao	0.53 [0.44, 0.61]	0.52 [0.43, 0.61]	0.56 [0.47, 0.65]	0.50 [0.41, 0.59]	0.51 [0.42, 0.60]	0.47 [0.40, 0.55]
cdpeo	0.55 [0.51, 0.59]	0.47 [0.41, 0.52]	0.56 [0.52, 0.60]	0.40 [0.36, 0.45]	0.51 [0.48, 0.55]	0.59 [0.55, 0.64]
covid19-ibo	0.56 [0.52, 0.60]	0.73 [0.69, 0.77]	0.81 [0.77, 0.84]	0.64 [0.60, 0.69]	0.75 [0.71, 0.79]	0.55 [0.52, 0.58]
ecp	0.61 [0.57, 0.66]	0.62 [0.57, 0.67]	0.83 [0.79, 0.87]	0.54 [0.49, 0.59]	0.80 [0.75, 0.85]	0.58 [0.54, 0.61]
emo	0.60 [0.55, 0.64]	0.74 [0.69, 0.78]	0.78 [0.74, 0.82]	0.62 [0.57, 0.66]	0.71 [0.66, 0.75]	0.54 [0.51, 0.57]
evi	0.65 [0.60, 0.70]	0.70 [0.66, 0.75]	0.70 [0.65, 0.75]	0.59 [0.54, 0.63]	0.50 [0.46, 0.55]	0.42 [0.38, 0.46]
falls	0.60 [0.55, 0.64]	0.62 [0.56, 0.68]	0.81 [0.76, 0.86]	0.56 [0.50, 0.62]	0.82 [0.77, 0.86]	0.61 [0.57, 0.66]
fo	0.44 [0.39, 0.50]	0.68 [0.63, 0.74]	0.49 [0.43, 0.55]	0.70 [0.64, 0.75]	0.54 [0.49, 0.59]	0.33 [0.27, 0.39]
gbm	0.61 [0.57, 0.66]	0.68 [0.63, 0.72]	0.73 [0.68, 0.77]	0.53 [0.49, 0.58]	0.62 [0.57, 0.66]	0.54 [0.50, 0.58]
gfvo	0.67 [0.63, 0.71]	0.72 [0.68, 0.75]	0.75 [0.71, 0.79]	0.55 [0.51, 0.59]	0.56 [0.51, 0.60]	0.48 [0.44, 0.52]
koro	0.63 [0.57, 0.68]	0.58 [0.52, 0.63]	0.70 [0.65, 0.75]	0.43 [0.38, 0.49]	0.59 [0.53, 0.64]	0.59 [0.54, 0.64]
lico	0.63 [0.58, 0.67]	0.65 [0.61, 0.70]	0.72 [0.67, 0.77]	0.52 [0.48, 0.57]	0.55 [0.51, 0.60]	0.49 [0.45, 0.53]
mamo	0.62 [0.57, 0.67]	0.71 [0.67, 0.76]	0.74 [0.69, 0.78]	0.61 [0.56, 0.66]	0.59 [0.54, 0.63]	0.46 [0.42, 0.50]
mpio	0.54 [0.49, 0.59]	0.75 [0.70, 0.79]	0.78 [0.74, 0.83]	0.74 [0.70, 0.78]	0.78 [0.73, 0.83]	0.36 [0.33, 0.40]
pizza	0.56 [0.49, 0.63]	0.65 [0.59, 0.72]	0.60 [0.53, 0.66]	0.56 [0.50, 0.63]	0.55 [0.49, 0.61]	0.46 [0.40, 0.53]
provo	0.68 [0.64, 0.72]	0.70 [0.66, 0.73]	0.75 [0.70, 0.79]	0.54 [0.50, 0.58]	0.53 [0.49, 0.57]	0.45 [0.42, 0.49]
qudt	0.57 [0.52, 0.62]	0.62 [0.55, 0.69]	0.95 [0.92, 0.98]	0.51 [0.45, 0.58]	0.96 [0.93, 0.99]	0.74 [0.70, 0.78]
trans	0.54 [0.48, 0.59]	0.61 [0.55, 0.66]	0.61 [0.55, 0.67]	0.57 [0.51, 0.63]	0.60 [0.55, 0.65]	0.49 [0.44, 0.54]
triage	0.63 [0.58, 0.68]	0.64 [0.59, 0.69]	0.70 [0.66, 0.75]	0.50 [0.45, 0.54]	0.54 [0.50, 0.59]	0.50 [0.46, 0.55]
vio	0.57 [0.50, 0.64]	0.61 [0.53, 0.68]	0.61 [0.54, 0.68]	0.54 [0.47, 0.61]	0.57 [0.50, 0.64]	0.47 [0.40, 0.53]
Overall	0.59 [0.58, 0.61]	0.65 [0.64, 0.67]	0.71 [0.70, 0.73]	0.56 [0.54, 0.57]	0.63 [0.61, 0.64]	0.51 [0.49, 0.52]

Table F.3: Results of the evaluation. IIC^+ is given as mean and 95% confidence interval in brackets.

$\lvert \mathrm{Inf}^+(\mathcal{O}) \rvert$ of	Weakening[a]	Weakening	MCS	Removal
admin	14090 [13871, 14316]	13997 [13800, 14209]	14052 [13815, 14308]	13867 [13679, 14069]
ahso	21480 [19911, 23104]	19894 [18449, 21376]	20508 [18976, 22047]	20036 [18630, 21451]
cdao	49753 [45987, 53861]	46405 [43664, 49283]	47751 [44809, 50910]	47929 [44768, 51439]
cdpeo	13520 [13473, 13583]	13459 [13432, 13489]	13535 [13484, 13594]	13460 [13436, 13489]
covid19-ibo	22924 [21923, 23936]	22432 [21494, 23376]	21905 [21020, 22821]	21452 [20508, 22384]
ecp	6781 [6616, 6942]	6416 [6271, 6553]	7047 [6875, 7215]	6374 [6232, 6508]
emo	24492 [23507, 25505]	23559 [22601, 24529]	22887 [21945, 23873]	22617 [21664, 23593]
evi	11838 [11295, 12386]	11753 [11176, 12327]	11366 [10820, 11904]	11854 [11293, 12416]
falls	684 [643, 728]	659 [620, 702]	684 [645, 725]	641 [604, 682]
fo	4571 [4278, 4874]	4625 [4327, 4933]	4434 [4144, 4736]	4514 [4223, 4819]
gbm	29672 [28862, 30465]	28591 [27757, 29415]	28745 [27974, 29501]	28158 [27319, 29021]
gfvo	21577 [21232, 21905]	20410 [19998, 20807]	19909 [19459, 20338]	20198 [19750, 20621]
koro	6923 [6656, 7223]	6233 [6141, 6328]	7074 [6783, 7395]	6189 [6111, 6273]
lico	41105 [39954, 42219]	39492 [38250, 40738]	39777 [38601, 40966]	39678 [38395, 40904]
mamo	17061 [16366, 17697]	16789 [16157, 17376]	15624 [14963, 16252]	15928 [15213, 16626]
mpio	2335 [2173, 2504]	2363 [2201, 2536]	2055 [1913, 2200]	2306 [2145, 2475]
pizza	93287 [88428, 98182]	88938 [83921, 93924]	87752 [83100, 92510]	86790 [81999, 91715]
provo	28025 [26972, 29128]	27473 [26431, 28590]	27144 [26053, 28257]	27542 [26450, 28687]
qudt	13218 [12711, 13775]	13087 [12612, 13624]	13426 [12886, 14034]	12990 [12537, 13505]
trans	8179 [8086, 8274]	8073 [7995, 8149]	8231 [8112, 8350]	8006 [7930, 8080]
triage	30818 [28930, 32803]	30248 [28318, 32291]	30628 [28759, 32641]	30495 [28592, 32548]
vio	26354 [24095, 28707]	23936 [21603, 26325]	23871 [21698, 26068]	23958 [21581, 26405]

Table F.4: The cardinality of the extended inferred class hierarchy is given as mean and 95% confidence interval in brackets. [a]Weakening based repair that keeps the reference ontology as static axioms that are not weakened or replaced.

		Time per weakening [ms]																	
		Caching using transitivity						Simple caching						No caching					
		1	5	10	20	50	100	1	5	10	20	50	100	1	5	10	20	50	100
FaCT++	admin	83.6	23.6	15.7	7.5	4.2	2.6	162.1	42.8	23.3	12.6	5.8	3.2	416.8	503.9	543.6	389.4	389.5	369.1
	cdpeo	99.6	52.2	37.5	17.1	8.9	5.4	227.6	81.0	44.8	24.0	11.4	6.4	429.3	437.3	540.7	413.8	397.8	409.3
	emo	72.2	52.3	26.5	15.0	8.3	4.7	83.8	33.4	25.9	20.4	11.8	6.9	113.4	164.3	131.1	114.0	110.9	108.0
	gbm	101.6	63.3	39.9	21.1	10.8	5.8	150.2	56.6	39.5	24.8	12.2	6.4	240.7	272.0	232.5	220.3	218.5	217.7
	gfvo	55.7	26.8	13.3	8.6	4.2	2.2	31.5	17.6	13.9	9.5	5.1	2.7	31.6	35.0	30.4	30.7	30.1	29.7
	koro	81.3	37.0	21.5	12.8	6.2	3.4	72.9	45.0	34.0	22.7	11.6	6.2	103.0	114.6	97.0	93.9	93.9	94.0
	mamo	45.7	24.8	13.3	6.8	3.3	1.6	33.8	20.2	14.6	8.8	4.5	2.4	48.8	51.0	40.7	40.4	40.6	42.1
	Overall	77.1	40.0	24.0	12.7	6.5	3.7	108.8	42.4	28.0	17.6	8.9	4.9	197.7	225.5	230.9	186.1	183.0	181.4
HermiT	admin	111.0	32.4	22.4	11.1			585.2	152.5	86.9	41.3			1469.9	1448.1	1519.6	1388.0		
	cdpeo	187.1	73.2	48.5	34.2			481.1	186.0	105.7	68.5			1017.3	894.3	984.6	975.8		
	emo	143.0	71.4	59.3	34.5			275.7	112.3	100.9	61.3			444.0	342.3	384.3	358.4		
	gbm	149.0	72.2	53.4	30.1			424.9	164.8	105.6	56.1			523.8	632.9	629.5	557.5		
	gfvo	90.1	40.4	29.3	16.8			89.2	54.0	46.0	28.9			115.2	87.7	97.8	81.5		
	koro	264.8	144.6	104.3	54.3			494.7	303.7	255.8	163.6			752.9	676.2	702.3	634.4		
	mamo	102.0	38.5	26.7	13.3			128.0	63.0	47.8	26.8			154.0	124.3	136.5	119.9		
	Overall	149.6	67.5	49.1	27.8			354.1	148.0	106.9	63.8			639.6	600.8	636.4	587.9		
JFact	admin	128.1	37.1	21.0	15.1			751.6	193.2	99.3	58.9			1877.9	1938.7	1806.7	2109.6		
	cdpeo	472.1	160.3	95.4	53.6			2404.9	775.5	428.3	222.5			4764.2	4193.6	4432.1	4657.4		
	emo	302.1	157.9	129.2	85.0			781.9	327.1	233.9	205.3			1170.6	1126.2	1068.0	1295.2		
	gbm	520.1	238.1	172.2	123.1			1834.8	701.8	447.9	335.0			2862.4	2878.8	2965.3	2981.6		
	gfvo	189.3	84.7	53.5	30.5			215.4	155.4	116.4	79.4			345.7	294.1	293.3	290.1		
	koro	286.4	123.8	71.2	41.0			595.5	333.6	228.6	148.4			681.2	763.0	696.3	678.9		
	mamo	181.3	66.8	39.0	21.7			234.2	132.4	89.7	55.4			328.3	296.1	298.7	291.1		
	Overall	297.1	124.1	83.1	52.8			974.1	374.1	234.9	157.8			1718.6	1641.5	1651.5	1757.7		
Openllet	admin	125.0	42.7	25.9	19.7			489.5	138.9	79.2	71.3			1026.0	1026.5	1028.9	1376.1		
	cdpeo	175.3	74.9	45.4	31.7			472.0	176.9	97.7	68.9			815.7	795.5	845.0	1025.4		
	emo	113.0	61.0	43.8	29.7			165.6	77.4	69.8	49.9			252.5	210.6	197.3	212.2		
	gbm	153.5	73.0	48.9	32.0			276.2	120.5	79.7	51.2			391.4	403.7	406.6	418.1		
	gfvo	50.4	23.8	15.4	11.9			47.8	26.3	17.8	13.5			54.6	37.9	34.1	37.5		
	koro	282.5	127.2	79.2	55.3			399.2	250.2	202.6	152.6			499.9	491.9	595.2	549.4		
	mamo	68.8	25.6	23.9	11.8			62.2	32.6	33.2	17.4			71.7	59.6	75.1	58.9		
	Overall	138.3	61.2	40.3	27.4			273.2	117.5	82.9	60.7			444.5	432.2	454.6	514.3		

Table F.5: Results of the evaluation of cache effectiveness. The mean execution time required for a single weakening is given.

	Reasoner calls per weakening																	
	Caching using transitivity						Simple caching						No caching					
	1	5	10	20	50	100	1	5	10	20	50	100	1	5	10	20	50	100
admin	1096	381	222	131	64	35	15138	4072	2115	1092	459	236	41105	42224	41288	41376	42663	41751
cdpeo	2110	869	522	301	136	75	13621	4936	2694	1436	608	312	29051	27289	28617	28818	28244	29315
emo	2652	1720	1355	864	463	258	7781	3436	2784	2023	1103	619	12524	12357	12134	12548	12462	12552
gbm	3019	1577	1074	674	309	168	12572	5065	3284	2057	956	503	19490	20945	21330	20581	20780	20801
gfvo	1737	891	575	335	151	80	2828	1975	1524	1043	539	293	4058	3973	4104	4012	3977	4003
koro	2006	952	591	344	157	80	5154	3046	2272	1465	721	382	7271	7618	7181	7223	7233	7422
mamo	1984	844	511	286	122	61	3557	2140	1488	943	449	234	5060	5011	5059	5037	4997	4998
Overall	2086	1033	693	419	200	108	8664	3524	2309	1437	691	368	16937	17059	17102	17007	17194	17263

Table F.6: Results of the evaluation of cache effectiveness. The mean number of reasoner calls required for a single weakening is given.

	Steps	Calls	Time [ms]	Failed
admin	13.3 [4.3, 25.0]	3199 [2619, 3929]	1834 [1142, 2869]	0% (2%)
ahso	13.6 [5.8, 25.2]	3489 [2981, 4136]	12082 [7920, 16710]	2% (28%)
cdao	3.9 [2.9, 5.2]	17109 [15581, 18690]	16868 [12180, 22423]	5% (51%)
cdpeo	3.0 [2.4, 4.0]	4359 [3903, 4850]	1375 [1283, 1525]	0% (0%)
covid19-ibo	2.1 [1.8, 2.5]	14377 [13976, 14791]	1698 [1644, 1757]	0% (13%)
ecp	7.4 [5.1, 10.0]	1686 [1435, 1975]	2912 [1840, 4494]	1% (15%)
emo	2.7 [2.2, 3.3]	19406 [18699, 20060]	7771 [4859, 11450]	0% (4%)
evi	91.9 [71.8, 114.0]	5577 [4316, 7016]	6751 [4733, 9214]	1% (64%)
falls	3.1 [2.4, 4.0]	812 [700, 944]	651 [609, 701]	0% (5%)
fo	10.4 [6.2, 16.1]	1095 [591, 1863]	1358 [820, 2235]	4% (63%)
gbm	3.5 [2.8, 4.1]	7726 [7355, 8097]	1793 [1733, 1862]	0% (0%)
gfvo	10.3 [7.6, 14.1]	3155 [2786, 3647]	1527 [1303, 1906]	0% (0%)
koro	5.6 [4.6, 6.8]	6353 [5763, 7006]	1406 [1275, 1554]	0% (0%)
lico	10.7 [8.6, 13.0]	4671 [4315, 5048]	2438 [2220, 2679]	0% (22%)
mamo	13.3 [10.7, 15.9]	2749 [2505, 3001]	1325 [1237, 1423]	0% (1%)
mpio	4.2 [3.5, 5.0]	1055 [1004, 1109]	699 [674, 727]	0% (30%)
pizza	28.4 [21.3, 36.8]	7180 [6203, 8261]	18099 [13518, 23582]	8% (59%)
provo	35.2 [27.1, 44.1]	2860 [2427, 3358]	4831 [3398, 6682]	2% (9%)
qudt	1.8 [1.5, 2.1]	6369 [6183, 6569]	5832 [3533, 8749]	1% (39%)
trans	2.8 [2.1, 3.6]	2191 [2035, 2356]	1063 [997, 1139]	0% (5%)
triage	34.0 [23.0, 47.2]	3186 [2690, 3785]	8790 [5447, 12864]	4% (53%)
vio	10.5 [7.4, 14.4]	7101 [6539, 7682]	12995 [8246, 18875]	2% (8%)

Table F.7: Results of the evaluation with respect to performance while repairing using axiom weakening, but keeping the axioms of the reference ontology unchanged during the repair. The number of weakening iterations, reasoner calls, and total repair time are given as sample mean, with a 95% confidence interval in brackets. The frequency of failed runs is shown as percentage of repairs by weakening that were started but not completed. In parentheses, the percentage of total failed runs, including those with timeout during generation of the inconsistent ontology. Note that repair using this method has only been attempted after the normal weakening based repair has been successful. This may skew the frequency of failures.