Freie Universität Bozen
Libera Università di Bolzano
Università Liedia de Bulsan

Fakultät für Ingenieurwesen
Facoltà di Ingegneria
Faculty of Engineering

Bachelor Thesis

Extending Axiom Weakening for Automated Repair of Ontologies 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, finance, and education. The introduction of the W3C recommended Web Ontology Language 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, ontologies become more susceptible to bugs, and it becomes harder to debug these defects. While debugging in software engineering has received much attention, tooling support for debugging ontologies remains limited. Moreover, in the context of the semantic web, automatic approaches to debugging ontologies are required for the combination of knowledge derived from independent sources. Axiom weakening has been proposed as a solution for fine-grained repair to inconsistent ontologies. This thesis presents an extension to the axiom weakening operator, in the $\mathcal{SROIQ}$ description logic, to cover a wider range of axiom types, including role inclusion axioms and role assertions. I show that the presented weakening operator retains desirable properties satisfied by previous approaches. The presented automated repair approach is experimentally evaluated against other repair approaches on a number of inconsistent ontologies. The paper compares the amount of information that the repair is able to retain relative to other repairs based on the inferred concept hierarchy. Further, the implementation of the presented axiom weakening operator in the popular ontology editor, Protégé, is discussed. The efficacy of integrating axiom weakening in the manual debugging process is demonstrated for inconsistent ontologies and unintended consequences in general.

Contents

Acknowledgements

Thanks Mami and Tati!

Chapter 1

Introduction

Chapter 2

Background and Related Work

2.1 Ontology Bugs

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

2.1.1 Categories of Bugs

Defects, in both software systems and ontologies, can be due to a number of different reasons. In [8] 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. Similarly, for software systems, these defects may be the result of programs that are not consistent with the grammar of the chosen programming language. These sorts of syntactic 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 1.

There may however be some additional restrictions on what constitutes a valid ontology or program that is not based solely on the grammatical rules. For ontologies, these might be for example the restrictions placed upon the form of the graph for a specific OWL profile. For programming languages, a similar restriction to this may be the requirement for definition before use or the presence of a type system[1]. These restrictions reduce the space of valid programs. Restrictions of this kind can often be much easier to violate and

[1] When viewed from a different perspective, a type error can also be seen as the unsatisfiability of (or ambiguity in) the type assertions. In this way, it is related to an inconsistency in the context of ontologies in the case of unsatisfiability (or missing inferences in the case of ambiguity).

harder to debug than the first kind of syntactic defects that is a violation of the grammar.

Example 2.

Semantic Defects

For ontologies, semantic defects, as defined in [8] 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.

Example 3.

A close analogy to these kinds of defects from the perspective of a software system is the raising of an error during the execution. An error is an indication of a defect in the software, and depending on tooling support they may be more or less difficult to understand and rectify.

Modelling Defects

Modelling defects are those defects that are not syntactically or semantically invalid. The presence of unintended inferences in an ontology is one such defect. These defects can also be of more stylistic nature. Redundancy or unused parts of the ontology may be considered as defects, since they do not add any knowledge to the ontology.

For software systems, modelling defects are bugs that do not cause any errors, but which produce undesired behaviour. An example for such a defect could be that the result of a calculation is wrong, or that the software includes security vulnerabilities. For software systems, there might be other non-functional requirements, that if not met constitute defects in the software. These may for example be unsatisfactory performance or unmaintainable code organization.

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 against some modelling defects, by expressing more explicitly the modellers/programmers intend. This can be done both for software systems and for ontologies.

2.1.2 Causes of Bugs

2.2 Ontologies in Description Logics

While ontologies can be represented using a number of different formalisms, for the 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 of the knowledge.

Description logics (DL) are often used for building ontologies. They encompass a family of related knowledge representation languages and are often fragments of FOL[2] with equality, as is the case with the description logics $\mathcal{SROIQ}$ 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 description logics have been developed for different applications, that feature varying levels of expressivity.

Description logics are also the basis of the *Web Ontology Language* (OWL) [4, 14], which is a World Wide Web Consortium (W3C) recommendation and is extensively used as part of the semantic web. While the OWL 2 language is based on $\mathcal{SROIQ}$, OWL 2 also defines three so-called profiles that are fragments of the full OWL 2 language that trade-off expressive power for more efficient reasoning [13, 14]. The OWL 2 DL profile, which is the most expressive of the profiles that is still decidable[3], is based on $\mathcal{SROIQ}$.

The following sections will introduce the description logic $\mathcal{SROIQ}$ and the OWL 2 language. The relation between the two will also be discussed. The description is also loosely based on the description in [16].

2.2.1 The $\mathcal{SROIQ}$ Description Logic

$\mathcal{SROIQ}$ Syntax

This section describes the syntax of the $\mathcal{SROIQ}$ description logic [6].

The *vocabulary*[4] $\mathrm{N} = \mathrm{N}_I \cup \mathrm{N}_C \cup \mathrm{N}_R$ of a $\mathcal{SROIQ}$ knowledge base is made up of three disjoint sets:

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

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

RBox The RBox $\mathcal{R}$ describes the relationship between different roles in the knowledge base. 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 names $r \in \mathrm{N}_R$, where r^- is called the *inverse role or* r. For convenience in the latter definitions, and to avoid roles like r^{--}, we

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

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

[4]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.

define a function Inv such that $\mathrm{Inv}(r) = r^-$ and $\mathrm{Inv}(r^-) = r$. We denote the set of all roles as $\mathbf{R} = \mathrm{N}_R \cup \{u\} \cup \{r^- \mid r \in \mathrm{N}_R\}$.

A *role inclusion axiom* (RIA) is a statement of the form $r_1 \circ \cdots \circ r_n \sqsubseteq r$ where $r, r_1, \ldots r_n \in \mathbf{R}$ 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 $\mathcal{R}_h$.

Roles can be partitioned into two disjoint sets, simple roles and non-simple roles. Intuitively, non-simple roles are those that are implied by the composition of two or more other roles. In order to preserve decidability, $\mathcal{SROIQ}$ requires that in parts of 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 $r_1 \circ \cdots \circ r_n \sqsubseteq r$ where $n > 1$ is non-simple,
- any role r that appears in a simple role inclusion $s \sqsubseteq r$ 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*. We denote the set of all non-simple roles with $\mathbf{R}^N$ and the set of simple roles with $\mathbf{R}^S = \mathbf{R} \setminus \mathbf{R}^N$.

Example 4.

There is an additional restriction that is placed upon the role hierarchy in a *SROIQ* knowledge base. The role hierarchy in $\mathcal{SROIQ}$ must be regular. A role hierarchy $\mathcal{R}_h$ is *regular* if there exists a strict partial order $\prec$ (that is, an irreflexive and transitive relation) on the set of roles $\mathbf{R}$, such that $s \prec r \iff \mathrm{Inv}(s) \prec r$ and $s \prec r \iff \mathrm{Inv}(s) \prec \mathrm{Inv}(r)$ for all roles r and s, and all RIA in $\mathcal{R}_h$ are $\prec$-regular. A RIA is defined to be *$\prec$-regular* if it is of one of the following forms:

- $r \circ r \sqsubseteq r$,
- $\mathrm{Inv}(r) \sqsubseteq r$,
- $r \circ s_1 \circ \cdots \circ s_n \sqsubseteq r$,
- $s_1 \circ \cdots \circ s_n \circ r \sqsubseteq r$, or
- $s_1 \circ \cdots \circ s_n \sqsubseteq r$,

such that $s_1, \ldots, s_n, r \in \mathbf{R}$ are roles, and s_i is simple or $s_i \prec r$ for all $i = 1, \ldots, n$.

This condition on the role hierarchy prevents cyclic definitions with role inclusion axioms that include role chains. These types of cyclic definition could otherwise lead to undecidability of the logic.

Example 5.

Example 6.

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 [6] in that it always allows simple roles on the left-hand side. Additionally, it is more permissive than stated in [16] in that it allows for inverse roles on the right-hand side.

The set of *role assertions* $\mathcal{R}_a$ is a finite set of statements with the form $\mathrm{Dis}(s_1, s_2)$ (*disjointness*) where s_1, and s_2 are simple roles in $\mathcal{R}_h$. In [6] 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 $\{r^- \sqsubseteq r\}$, $\{\mathrm{Dis}(r, r^-)\}$, $\{r \circ r \sqsubseteq r\}$, $\{r' \sqsubseteq r, \top \sqsubseteq \exists r'.\mathrm{Self}\}$, and $\{\top \sqsubseteq \neg\exists r.\mathrm{Self}\}$ respectively. Note that the asymmetry assertion requires a simple role, and that r' in the case of reflexivity must be a role name not otherwise used in the ontology [5].

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 $C \in \mathrm{N}_C$ are concept, called *atomic concepts*,
- all finite subsets of individual names $\{a_1, \ldots, a_n\} \subseteq \mathrm{N}_I$ 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 $r \in \mathbf{R}$ a (possible non-simple) role, then $\exists r.C$ (*existential quantification*) and $\forall r.C$ (*universal quantification*) are also concepts, and
- if C is a concept, $s \in \mathbf{R}^S$ a simple role and $n \in \mathbb{N}_0$ a non-negative number, then $\exists r.\mathrm{Self}$ (*self restriction*), $\leq ns.C$ (*at-most restriction*), and $\geq ns.C$ (*at-least restriction*) are concepts, the last two may together be referred to as *qualified number restrictions.*

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

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 $a \in \mathrm{N}_I$,
- $r(a, b)$ (*role assertion*) or $\neg r(a, b)$ (*negative role assertion*) for some role $r \in \mathbf{R}$ and individual names $a, b \in \mathrm{N}_I$, or

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

- $a = b$ (*equality*) or $a \neq b$ (*inequality*) for some individual names $a \in \mathrm{N}_I$.

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

$\mathcal{SROIQ}$ Semantics

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

- for each individual name $a \in \mathrm{N}_I$ to an element $a^\mathcal{I} \in \Delta^\mathcal{I}$ in the domain,
- for each concept name $C \in \mathrm{N}_C$ to a subset $C^\mathcal{I} \subseteq \Delta^\mathcal{I}$ of the domain, and
- for each role name $r \in \mathrm{N}_R$ to a relation $r^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$ over the domain.

An interpretation maps the universal role u to $u^\mathcal{I} = \Delta^\mathcal{I} \times \Delta^\mathcal{I}$. We extend the interpretation function to operate over inverse roles such that $(r^-)^\mathcal{I}$ contains exactly those elements $\langle \delta_1, \delta_2 \rangle$ for which $\langle \delta_2, \delta_1 \rangle$ is contained in $r^\mathcal{I}$, that is $(r^-)^\mathcal{I} = \{\langle \delta_1, \delta_2 \rangle \mid \langle \delta_2, \delta_1 \rangle \in r^\mathcal{I}\}$. 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 $\top^\mathcal{I} = \Delta^\mathcal{I}$.
- The bottom concept is true for no individual, hence $\bot^\mathcal{I} =$ where represents the empty set.
- Nominal concepts contain exactly the specified individuals, that is $\{a_1, \ldots, a_n\}^\mathcal{I} = \{a_1^\mathcal{I}, \ldots, a_n^\mathcal{I}\}$.
- $\neg C$ yields the complement of the extension of C, thus $(\neg C)^\mathcal{I} = \Delta^\mathcal{I} \setminus C^\mathcal{I}$.
- $C \sqcup D$ denotes all individuals that are in either the extension of C or in that of D, hence $(C \sqcup D)^\mathcal{I} = C^\mathcal{I} \cup D^\mathcal{I}$.
- $C \sqcap D$ 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 $(C \sqcap D)^\mathcal{I} = C^\mathcal{I} \cap D^\mathcal{I}$.
- $\exists r.C$ holds for all individuals that are connected by some element in the extension of r to an individual in the extension of C, formally $(\exists r.C)^\mathcal{I} = \{\delta_1 \in \Delta^\mathcal{I} \mid \exists \delta_2 \in \Delta^\mathcal{I} \,.\, \langle \delta_1, \delta_2 \rangle \in r^\mathcal{I} \wedge \delta_2 \in C^\mathcal{I}\}$.
- $\forall r.C$ 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 $(\forall r.C)^\mathcal{I} = \{\delta_1 \in \Delta^\mathcal{I} \mid \forall \delta_2 \in \Delta^\mathcal{I} \,.\, \langle \delta_1, \delta_2 \rangle \in r^\mathcal{I} \rightarrow \delta_2 \in C^\mathcal{I}\}$.
- $\exists r.\mathrm{Self}$ indicates all individuals that the extension of r connects to themselves, hence we let $(\exists r.\mathrm{Self})^\mathcal{I} = \{\delta \in \Delta^\mathcal{I} \mid \langle \delta, \delta \rangle \in r^\mathcal{I}\}$.

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

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

Satisfaction of Axioms The purpose of the (extended) interpretation function is mainly to determine satisfaction of axioms. We define in the following when an axiom α is true, or holds, in a specific interpretation $\mathcal{I}$. If this is the case, the interpretation $\mathcal{I}$ satisfies α, written $\mathcal{I} \models \alpha$. If an interpretation $\mathcal{I}$ satisfies an axiom α, we also say that $\mathcal{I}$ is a model of α.

- A role inclusion axiom $s_1 \circ \cdots \circ s_n \sqsubseteq r$ holds in $\mathcal{I}$ if and only if for each sequence $\delta_1, \ldots, \delta_{n+1} \in \Delta^{\mathcal{I}}$ for which $\langle\delta_i, \delta_{i+1}\rangle \in s_i^{\mathcal{I}}$ for all $i = 1, \cdots, n$, also $\langle\delta_1, \delta_n\rangle \in r^{\rangle}$ is satisfied. Equivalently, we can write $\mathcal{I} \models s_1 \circ \cdots \circ s_n \sqsubseteq r \iff s_1^{\mathcal{I}} \circ \cdots \circ s_n^{\mathcal{I}} \subseteq r^{\mathcal{I}}$ where $\circ$ denotes the composition of the relations.

- A role reflexivity axiom $\mathrm{Ref}(r)$ hold iff for each element of the domain $\delta \in \Delta^{\mathcal{I}}$ the condition $\langle\delta, \delta\rangle \in r^{\mathcal{I}}$ is satisfied. In other words, $\mathcal{I} \models \mathrm{Ref}(r) \iff \{\langle\delta, \delta\rangle \mid \delta \in \Delta^{\mathcal{I}}\} \subseteq r^{\mathcal{I}}$.

- A role asymmetry axioms $\mathrm{Asy}(r)$ is holds $\mathcal{I} \models \mathrm{Asy}(r)$ iff $\langle\delta_1, \delta_2\rangle \in r^{\mathcal{I}}$ implies that $\langle\delta_2, \delta_1\rangle \notin r^{\mathcal{I}}$.

- A role disjointness axiom $\mathrm{Dis}(s, r)$ hold iff the extensions of r and s are disjoint, formally $\mathcal{I} \models \mathrm{Dis}(s, r) \iff s^{\mathrm{I}} \cap r^{\mathrm{I}} =$.

- A general concept inclusion axiom $C \sqsubseteq D$ is true iff the extension of C is fully contained in the extension of D, hence $\mathcal{I} \models C \sqsubseteq D \iff C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$.

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

- A role assertion $r(a, b)$ holds iff the individuals denoted by the name a and b are connected in the extension of r, thus $\mathcal{I} \models r(a, b) \iff \langle a^{\mathcal{I}}, b^{\mathcal{I}}\rangle \in r^{\mathcal{I}}$.

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

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

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

We say a set of axioms holds in an interpretation $\mathcal{I}$ iff every axiom of the set hold in $\mathcal{I}$. Accordingly, $\mathcal{I}$ satisfies a knowledge base $\mathcal{KB}$, written $\mathcal{I} \models \mathcal{KB}$, iff $\mathcal{I}$ satisfies every axiom $\alpha \in \mathcal{KB}$ of the knowledgeable, i.e., $\mathcal{I} \models \mathcal{KB} \iff \forall\alpha \in \mathcal{KB} \,.\, \mathcal{I} \models \alpha$. If $\mathcal{I}$ satisfies $\mathcal{KB}$, we say $\mathcal{I}$ is a model of $\mathcal{KB}$.

Reasoning tasks

In general, logic-based knowledge representation is useful for the ability to perform reasoning task on knowledge bases. There are a number of reasoning tasks that can be performed by a reasoner in description logics. In this section, we will take a look at three basic reasoning tasks, and how they can be reduced to each other. While there exists other reasoning task, this section will focus on *knowledge base satisfiability, axiom entailment, and concept satisfiability.*

Knowledge base satisfiability A knowledge $\mathcal{KB}$ base is satisfiable iff there exists a model $\mathcal{I} \models \mathcal{KB}$ for $\mathcal{KB}$. Otherwise, the knowledge base is called unsatisfiable, inconsistent, or contradictory. As discussed in section 2.1, an inconsistent knowledge base can be a sign of modelling errors. An inconsistent knowledge base entailed every statement, and as such all information extracted from it is useless. Therefore, an unsatisfiable knowledge base is generally undesirable. Furthermore, both the task of deciding concept satisfiability and axiom entailment can be reduced to deciding knowledge base consistency.

Axiom entailment An axiom α is entailed by $\mathcal{KB}$ if every model $\mathcal{I} \models \mathcal{KB}$ of the knowledge base also satisfies the axiom $\mathcal{I} \models \alpha$. We also write this as $\mathcal{KB} \models \alpha$ and say that α is a consequence of $\mathcal{KB}$. Deciding axiom entailment is an important task in order to derive new information from the collected knowledge. If α is entailed by the empty knowledge base, α is said to be a *tautology.* Further, the *set of consequences* of $\mathcal{KB}$ if the set of all axioms which are entailed by $\mathcal{KB}$, we write $\mathrm{Con}(\mathcal{KB}) = \{\alpha \mid \mathcal{KB} \models \alpha\}$. 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 for the satisfiability of a modified knowledge base. This is achieved by using an axiom β that imposes the opposite restriction to α, to be more precise, for all interpretations $\mathcal{I} \models \alpha \iff \mathcal{I} \not\models \beta$. If α is entailed by $\mathcal{KB}$, it must hold in every model of $\mathcal{KB}$, hence β must not hold in any model. It follows that the extended knowledge base $\mathcal{KB} \cup \{\,\beta\,\}$ has no new model, and is therefore unsatisfiable. We can consequently solve the axiom entailment problem by testing for satisfiability of a modified knowledge base, 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. Section 2.2.1 shows the correspondence for every type of $\mathcal{SROIQ}$ axiom.

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

Like axiom entailment, concept satisfiability can be reduced to knowledge base satisfiability. If a concept is unsatisfiable, every model $\mathcal{I} \models \mathcal{KB}$ maps the concept to the empty set, that is $C^{\mathcal{I}} =$. Since the other direction is trivial, we can rewrite this as $C^{\mathcal{I}} \subseteq$. It follows that since $\bot^{\mathcal{I}} =$ the every such model

α	B
$s_1 \circ \dots \circ s_n \sqsubseteq r$	$s_1(a_1, a_2)$, ..., $s_n(a_n, a_{n+1})$, and $\neg r(a_1, a_{n+1})$
$\mathrm{Dis}(s, r)$	$s(a, b)$ 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 2.1: The axioms in B together have the "opposite" meaning of those in α. This means, checking entailment of α with respect to $\mathcal{KB}$, is equivalent to checking unsatisfiability of $\mathcal{KB} \cup B$. a, a_i, and b are assumed to not appear in $\mathcal{KB}$.

satisfies $\mathcal{I} \models C \sqsubseteq \bot$, meaning $\mathcal{KB} \models C \sqsubseteq \bot$. We conclude that we can test for unsatisfiability of a concept C by checking for entailment of $C \sqsubseteq \bot$.

2.2.2 The Web Ontology Language

2.3 Repairing Ontologies

As established in section 2.1, 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.

2.3.1 Basic Definitions

We define a repair as proposed in [1]. It is assumed, as is the case for most description logics, that there exists a monotone consequence operator $\models$ such that for any 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 be erroneous. This separation is useful for example if the static part of the ontology is hand-crafted, while the refutable part is automatically generated. Similarly, it is applicable in case multiple ontologies are combined, and some sources are seen as less trustworthy than others.

Definition: 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. In contrast, generally an optimal repair does not need to exist.

Example 7.

It should be noted also that there exists an infinite number of possible repairs, as adding tautologies to a repair will yield another valid repair. In the case that we are interested in making an inconsistent ontology consistent, we can use as α an 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 $\mathcal{O}$.

In contrast, the classical approach to repair consists of locating and removing problematic axioms. 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: 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. It follows 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 $\mathcal{O}^{(1)}, \mathcal{O}^{(2)}, \dots$ such that $\mathrm{Con}(\mathcal{O}^{(i)}) \subset \mathrm{Con}(\mathcal{O}^{(i+1)})$.

2.3.2 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 [15].

Definition: 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 in $\mathcal{O}$ is a minimal subset $J \subseteq \mathcal{O}_r$ such that $J \cup \mathcal{O}_s \models \alpha$. Given the set of all justifications $J_1, \dots, J_n$ for α in $\mathcal{O}$, a *hitting set* H for these justifications is a set of axioms such that $H \cap J_i \neq$ for $i = 1, \dots, n$. H is a *minimal hitting* set if it does not strictly contain another hitting set.

Example 8.

Justifications are necessarily non-empty since $\mathcal{O}_s \not\models \alpha$, and therefore hitting sets and minimal hitting sets always exist. Given any minimal hitting set H for the justification $J_1, \dots, J_n$ 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 [9, 17, 18] that compute justifications by repeatedly making calls to pre-existing highly-optimized reasoners. These may however, in the worst-case, need to make an exponential number of calls to the reasoner. Nevertheless, in practice they may often be fast enough, as for example the hitting set tree base algorithm presented in [9], which conveniently can compute both all justifications and all hitting sets. There exist also glass-box approach to computing justifications [9], 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 subsets over monotone predicate problem, such as the QuickXplain algorithm [7] or a progression-based algorithm [12] may be used. A subset of all such sets can be found efficiently using the MergeXplain algorithm [19]. Of course, computing all minimal correction subsets directly is also possible, using similar algorithms to the ones used for computing all justifications [11].

More Gentle Repairs

While the classical approach is sufficient to guarantee finding a repair of the ontology, it can lead to information loss. That is, the repaired ontology might be missing some consequences of the original ontology that were actually desirable.

Example 9.

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.

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 contain less information, and therefore the removal of axioms can be more granular relative to the unmodified ontology. In [5] the authors propose a structural transformation, that replaces axioms with a set of weaker axioms that are semantically equivalent.

Example 10.

Another approach to repairing ontologies more gently that has been proposed in the literature is using *axiom weakening* [20, 3, 2, 1, 10]. Instead of removing axioms, they are replaced with weaker axioms. In [10] the authors show a method for pinpointing the causes for unsatisfiability a within axioms, and propose a way of weakening axioms guided by this information. The authors of [1] show general theoretical results for repair using axiom weakening, and propose a concrete weakening relation for the $\mathcal{EL}$ description logics. They further show that the repair algorithm using the proposed axiom weakening terminates in at most an exponential number of weakening steps. [20] presents the repair of inconsistent ontologies using axiom weakening with the help of a refinement operator. This approach is extended in [3, 2] to cover more expressive description logics and a proof of almost sure termination.

2.4 Axiom Weakening

2.5 Problems of Expressivity

Chapter 3

Theoretical Foundations

3.1 RBox weakening

3.2 Repair using Heuristic Search

Chapter 4

Implementation

4.1 Implementation of $\mathcal{SROIQ}$ Weakening

4.2 Axiom Weakening in Protégé

Chapter 5

Experiments and Evaluation

Chapter 6

Conclusion and Future Work

Bibliography

[1] Franz Baader et al. "Making repairs in description logics more gentle". In: *Sixteenth International Conference on Principles of Knowledge Representation and Reasoning*. 2018.

[2] Roberto Confalonieri et al. "Irresistible Refinement Operators for Expressive Description Logics". Unpublished manuscript. 2022.

[3] Roberto Confalonieri et al. "Towards even more irresistible axiom weakening". In: (2020).

[4] Pascal Hitzler et al. "OWL 2 web ontology language primer". In: *W3C recommendation* 27.1 (2009), p. 123.

[5] 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.

[6] Ian Horrocks, Oliver Kutz, and Ulrike Sattler. "The Even More Irresistible SROIQ." In: *Kr* 6 (2006), pp. 57–67.

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

[8] Aditya Kalyanpur et al. "Debugging unsatisfiable classes in OWL ontologies". In: *Journal of Web Semantics* 3.4 (2005), pp. 268–293.

[9] Aditya Kalyanpur et al. "Finding all justifications of OWL DL entailments". In: *ISWC/ASWC* 4825 (2007), pp. 267–280.

[10] Joey Sik Chun Lam et al. "A fine-grained approach to resolving unsatisfiable ontologies". In: *Journal on Data Semantics X*. Springer. 2008, pp. 62–95.

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

[12] 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.

[13] Boris Motik et al. "OWL 2 web ontology language profiles". In: *W3C recommendation* 27.61 (2009).

[14] Boris Motik et al. "OWL 2 web ontology language: Structural specification and functional-style syntax". In: *W3C recommendation* 27.65 (2009), p. 159.

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

[16] 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.

[17] Stefan Schlobach, Ronald Cornet, et al. "Non-standard reasoning services for the debugging of description logic terminologies". In: *Ijcai*. Vol. 3. 2003, pp. 355–362.

[18] Stefan Schlobach et al. "Debugging incoherent terminologies". In: *Journal of Automated Reasoning* 39 (2007), pp. 317–349.

[19] 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.

[20] Nicolas Troquard et al. "Repairing ontologies via axiom weakening". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1. 2018.

Appendix A

Correctness of Axiom Weakening

Lemma 1. *For all $r' \in \mathrm{UpCover}_{\mathcal{O}}(r)$ and every model M of $\mathcal{O}$, $r^M \subseteq r'^M$.*

Proof. Following the definition of $\mathrm{UpCover}_{\mathcal{O}}(\cdot)$, $r \sqsubseteq_{\mathcal{O}} r'$ holds and by definition $r^M \subseteq r'^M$ must therefore be true in every model. □

Lemma 2. *For all $r' \in \mathrm{DownCover}_{\mathcal{O}}(r)$ and every model M of $\mathcal{O}$, $r'^M \subseteq r^M$.*

Proof. Following the definition of $\mathrm{DownCover}_{\mathcal{O}}(\cdot)$, $r' \sqsubseteq_{\mathcal{O}} r$ holds and by definition $r'^M \subseteq r^M$ must therefore be true in every model. □

Lemma 3. *The RBox $\mathcal{R}'$, obtained by replacing any RIA $\phi = s_1 \circ \cdots \circ s_n \sqsubseteq r$ in the regular RBox $\mathcal{R}$ with a RIA $\phi' = s_1 \circ \cdots \circ s_n \sqsubseteq r'$ where $r \sqsubseteq_{\mathcal{O}} r'$, is entailed by $\mathcal{R}$.*

Proof. Take an arbitrary model M of $\mathcal{R}$. All axioms in $\mathcal{R}'$ that are also in $\mathcal{R}$ are satisfied by M because M satisfies all axioms in $\mathcal{R}$. We know that $r^M \subseteq r'^M$. Since M satisfies all axioms in $\mathcal{R}$, it satisfies ϕ meaning $s_1^M \circ \cdots \circ s_n^M \subseteq r^M$. Using the transitivity of the subset relation on sets, we obtain $s_1^M \circ \cdots \circ s_n^M \subseteq r'^M$. By definition, it follows that M satisfies ϕ'. Since all other axioms of $\mathcal{R}'$ are also in $\mathcal{R}$, M satisfies all axioms of $\mathcal{R}'$ and is therefore a model for $\mathcal{R}'$. □

Lemma 4. *The RBox $\mathcal{R}'$, obtained by replacing any RIA $\phi = s_1 \circ \cdots \circ s_i' \circ \cdots \circ s_n \sqsubseteq r$ in the regular RBox $\mathcal{R}$ with a RIA $\phi' = s_1 \circ \cdots \circ s_i' \circ \cdots \circ s_n \sqsubseteq r$ where $s_i' \sqsubseteq_{\mathcal{O}} s_i$, is entailed by $\mathcal{R}$.*

Proof. Take an arbitrary model M of $\mathcal{R}$. All axioms in $\mathcal{R}'$ that are also in $\mathcal{R}$ are satisfied by M because M satisfies all axioms in $\mathcal{R}$. We know that $s_i'^M \subseteq s_i^M$. Since M satisfies all axioms in $\mathcal{R}$, it satisfies ϕ meaning $s_1^M \circ \cdots \circ s_i^M \circ \cdots \circ s_n^M \subseteq r^M$. From $s_i'^M \subseteq s_i^M$ it follows that $s_1^M \circ \cdots \circ s_i'^M \circ \cdots \circ s_n^M \subseteq s_1^M \circ \cdots \circ s_i^M \circ \cdots \circ s_n^M$. Using transitivity of the subset relation on sets, we obtain $s_1^M \circ \cdots \circ s_i'^M \circ \cdots \circ s_n^M \subseteq r^M$. By definition, it follows that M satisfies ϕ'. Since all other axioms of $\mathcal{R}'$ are also in $\mathcal{R}$, M satisfies all axioms of $\mathcal{R}'$ and is therefore a model for $\mathcal{R}'$. □

Lemma 5. *The RBox $\mathcal{R}'$, obtained by replacing any role assertion $\phi = \mathrm{Dis}(s_1, s_2)$ in the regular RBox $\mathcal{R}$ with a RIA $\phi' = \mathrm{Dis}(s_1', s_2')$ where $s_1' \sqsubseteq_{\mathcal{O}} s_1$ and $s_2' \sqsubseteq_{\mathcal{O}} s_2$, is entailed by $\mathcal{R}$.*

Proof. Take an arbitrary model M of $\mathcal{R}$. All axioms in $\mathcal{R}'$ that are also in $\mathcal{R}$ are satisfied by M because M satisfies all axioms in $\mathcal{R}$. We know that $s_1'^M \subseteq s_1^M$ and $s_2'^M \subseteq s_2^M$. Since M satisfies all axioms in $\mathcal{R}$, it satisfies ϕ meaning $s_1^M \cap s_2^M =$. We know that $s_1'^M \cap s_2'^M \subseteq s_1^M \cap s_2^M =$. By definition, it follows that M satisfies ϕ'. Since all other axioms of $\mathcal{R}'$ are also in $\mathcal{R}$, M satisfies all axioms of $\mathcal{R}'$ and is therefore a model for $\mathcal{R}'$. □

Lemma 6. *The RBox $\mathcal{R}'$, obtained by replacing any simple RIA $\phi = s \sqsubseteq r$ in the regular RBox $\mathcal{R}$ for which s is simple with a RIA $\phi' = s \sqsubseteq r'$ where $r' \in \mathbf{R}$, is regular.*

Proof. Since $\mathcal{R}$ is regular, there exist a relation $\prec$ that satisfies the necessary conditions. All axioms in $\mathcal{R}'$ that are also in $\mathcal{R}$ are $\prec$-regular. It follows that since s is simple, the new RIA is $\prec$-regular. Since $\mathcal{R}'$ contains no other axioms, all axioms in $\mathcal{R}'$ are $\prec$-regular. $\mathcal{R}'$ is regular since $\prec$ is an admissible regular order. □

Lemma 7. *The RBox $\mathcal{R}'$, obtained by replacing any RIA $\phi = s_1 \circ \cdots \circ s' \circ \cdots \circ s_n \sqsubseteq r$ in the regular RBox $\mathcal{R}$ with a RIA $\phi' = s_1 \circ \cdots \circ s' \circ \cdots \circ s_n \sqsubseteq r$ where s' is simple, is regular.*

Proof. Since $\mathcal{R}$ is regular, there exist a relation $\prec$ that satisfies the necessary conditions. All axioms in $\mathcal{R}'$ that are also in $\mathcal{R}$ are $\prec$-regular. We handle each case of ϕ separately:

(i) $\phi = r \circ r \sqsubseteq r$: $r \circ s' \sqsubseteq r$ ($s' \circ r \sqsubseteq r$) is $\prec$-regular because the first (last) roles in the left-hand side are equal to the right-hand side and s' is simple.

(ii) $\phi = \mathrm{Inv}(r) \sqsubseteq r$: $s' \sqsubseteq r$ is $\prec$-regular because s' is simple.

(iii) $\phi = r \circ s_1 \circ \cdots \circ s \circ \cdots \circ s_n \sqsubseteq r$ ($\phi = s_1 \circ \cdots \circ s \circ \cdots \circ s_n \circ r \sqsubseteq r$): $r \circ s_1 \circ \cdots \circ s' \circ \cdots \circ s_n \sqsubseteq r$ ($s_1 \circ \cdots \circ s' \circ \cdots \circ s_n \circ r \sqsubseteq r$) is $\prec$-regular because since the original axiom is $\prec$-regular, we know that s_i is simple or $s_i \prec r$ for all $i = 1, \ldots, n$. Additionally, the first (last) role on the left-hand side is equal to the right-hand side and s' is simple.

(iv) $\phi = s_1 \circ \cdots \circ s \circ \cdots \circ s_n \sqsubseteq r$: $s_1 \circ \cdots \circ s' \circ \cdots \circ s_n \sqsubseteq r$ is $\prec$-regular because s' is simple and since the original axiom is $\prec$-regular, we know that s_i is simple or $s_i \prec r$ for all $i = 1, \ldots, n$.

We conclude that the new axiom is $\prec$-regular. Since $\mathcal{R}'$ contains no other axioms, all axioms in $\mathcal{R}'$ are $\prec$-regular. $\mathcal{R}'$ is regular since $\prec$ is an admissible regular order. □

Lemma 8. *The RBox $\mathcal{R}'$ is obtained by replacing any simple RIA $\phi = s \sqsubseteq r$ in the regular RBox $\mathcal{R}$ for which s is simple with a RIA $\phi' = s \sqsubseteq r'$ where $r' \in \mathbf{R}$. All roles that are simple in $\mathcal{R}$ are also simple in $\mathcal{R}'$.*

Proof. Let t be an arbitrary role that is simple in $\mathcal{R}$. t is simple in $\mathcal{R}'$ because:

(i) $t \neq u$, otherwise t would not be simple in $\mathcal{R}$.

(ii) t does not appear on the right-hand side of a complex RIA. The only new axiom in $\mathcal{R}'$ is ϕ'. ϕ' is a simple RIA.

(iii) t does not appear in a simple RIA where the left-hand side is non-simple. The only new axiom in $\mathcal{R}'$ is ϕ'. The left-hand side of ϕ', s, is simple.

(iv) $\mathrm{Inv}(t)$ is simple because it is simple in $\mathcal{R}$ and the above three points hold also for $\mathrm{Inv}(t)$.

Since t is an arbitrary simple role from $\mathcal{R}$, all simple roles in $\mathcal{R}$ are simple in $\mathcal{R}'$. □

Lemma 9. *The RBox $\mathcal{R}'$ is obtained by replacing any RIA $\phi = s_1 \circ \cdots \circ s' \circ \cdots \circ s_n \sqsubseteq r$ in the regular RBox $\mathcal{R}$ with a RIA $\phi' = s_1 \circ \cdots \circ s' \circ \cdots \circ s_n \sqsubseteq r$ where s' is simple. All roles that are simple in $\mathcal{R}$ are also simple in $\mathcal{R}'$.*

Proof. Let t be an arbitrary role that is simple in $\mathcal{R}$. t is simple in $\mathcal{R}'$ because:

1. $t \neq u$, otherwise t would not be simple in $\mathcal{R}$.
2. t does not appear on the right-hand side of a complex RIA. The only new axiom in $\mathcal{R}'$ is ϕ'. If ϕ' is a complex RIA and $t = r$, then t would not be simple in $\mathcal{R}$ because it is on the right-hand side of the complex RIA ϕ. This contradicts our assumption that t is simple.
3. t does not appear in a simple RIA where the left-hand side is non-simple. The only new axiom in $\mathcal{R}'$ is ϕ'. The left-hand side of ϕ', s', is simple.
4. $\mathrm{Inv}(t)$ is simple because it is simple in $\mathcal{R}$ and the above three points hold also for $\mathrm{Inv}(t)$.

Since t is an arbitrary simple role from $\mathcal{R}$, all simple roles in $\mathcal{R}$ are simple in $\mathcal{R}'$. □

Theorem 1. *The set of well-formed $\mathcal{SROIQ}$ ontologies is closed under axiom weakening. That is, given a valid $\mathcal{SROIQ}$ ontology $\mathcal{O}$ with a regular RBox $\mathcal{R}$, the ontology $\mathcal{O}' = \mathcal{O} \setminus \{\alpha\} \cup \{\alpha'\}$ obtained by replacing any axiom $\alpha \in \mathcal{O}$ with a weakening $\alpha' \in g_{\mathcal{O}}(a)$ is also a valid $\mathcal{SROIQ}$ ontology.*

Proof. We must show that the axioms resulting from weakening are syntactically correct, that no non-simple role is used in disjoint role axioms, cardinality restrictions, or self restrictions, and that the resulting RBox is regular.

Regularity of the RBox follows from lemma 6 and lemma 7, since those are the only types of transformations performed on the RIAs, and regularity depends only on the RIAs. Similarly, it follows from lemma 8 and lemma 9 that all roles simple in $\mathcal{O}$ are also simple in $\mathcal{O}'$. This means all cardinality or self restriction for which the role is not changed, still use a simple role.

Numbers and roles in the axioms are replaced, if at all, always with either a number or role respectively. This is ensured by the definition of the upward and downward cover. The covers return only non-negative numbers, roles that appear in the signature of the ontology, or the inverse of those roles. Since all roles returned by the upward and downward covers are simple in $\mathcal{O}$ and therefore still simple in $\mathcal{O}'$, it is not possible to replace a simple role with a non-simple role in a disjoint role axiom.

Concepts are replaced always with another valid concept, since the refinement operator, given a well-formed concept, returns only well-formed concept expressions. By induction, the refinement operator returns either:

(i) The result of the upward or downward covers. These are by definition subconcepts of the original ontology, which, since it appears in the original ontology, must be well-formed.

(ii) An expression in which one part of the input is replaced:

- A number or role is replaced with a role from the upward or downward cover. As mentioned above, the covers return only respectively valid numbers or roles. Since all roles returned are simple in $\mathcal{O}'$ it is not possible that a non-simple role is used in a cardinality or self restriction.
- A concept is replaced with the output of applying the refinement operator to the concept, that it replaces. Since the concept that is replaced is in the original ontology it is well-formed, and the replacement is a well-formed concept expression given our inductive hypothesis.

$\square$

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 confuting 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 11. OWL has an axiom 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 $i \neq j = 1, \ldots, n$. One reasonable approach to weakening the OWL axiom is to replace any of the classes C_i with a more specific class $C_i' \in \rho_{\mathcal{O}}(C_i)$. 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 12. The OWL axiom 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 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, $\mathrm{UpCover}_{\mathcal{O}}$ and $\mathrm{DownCover}_{\mathcal{O}}$, are used. We define the refinement operator $\zeta_{\uparrow,\downarrow}$ for OWL 2 DL as follows:

$$\zeta_{\uparrow,\downarrow}(A) = \uparrow(A) \qquad \text{for } A \in \mathrm{N}_c \cup \mathbf{R} \cup \{\top, \bot\}$$

[1] In the rest of this section, the OWL 2 functional syntax (cf. [14]) will be used.

$$
\begin{aligned}
\zeta_{\uparrow,\downarrow}(\text{ObjectComplementOf}(C)) = {} & \uparrow(\text{ObjectComplementOf}(C)) \\
& \cup \{\text{ObjectComplementOf}(C') \mid C' \in \zeta_{\downarrow,\uparrow}(C)\} \\
\zeta_{\uparrow,\downarrow}(\text{ObjectIntersectionOf}(C_1,\dots,C_n)) = {} & \uparrow(\text{ObjectIntersectionOf}(C_1,\dots,C_n)) \\
& \cup \bigcup_{i=1}^{n} \{\text{ObjectIntersectionOf}(C_1,\dots,C_i',\dots C_n) \mid C_i' \in \zeta_{\uparrow,\downarrow}(C_i)\} \\
\zeta_{\uparrow,\downarrow}(\text{ObjectUnionOf}(C_1,\dots,C_n)) = {} & \uparrow(\text{ObjectUnionOf}(C_1,\dots,C_n)) \\
& \cup \bigcup_{i=1}^{n} \{\text{ObjectUnionOf}(C_1,\dots,C_i',\dots C_n) \mid C_i' \in \zeta_{\uparrow,\downarrow}(C_i)\} \\
\zeta_{\uparrow,\downarrow}(\text{ObjectAllValuesFrom}(r,C)) = {} & \uparrow(\text{ObjectAllValuesFrom}(r,C)) \\
& \cup \{\text{ObjectAllValuesFrom}(r',C) \mid r' \in \zeta_{\downarrow,\uparrow}(r)\} \\
& \cup \{\text{ObjectAllValuesFrom}(r,C') \mid C' \in \zeta_{\uparrow,\downarrow}(C)\} \\
\zeta_{\uparrow,\downarrow}(\text{ObjectSomeValuesFrom}(r,C)) = {} & \uparrow(\text{ObjectSomeValuesFrom}(r,C)) \\
& \cup \{\text{ObjectSomeValuesFrom}(r',C) \mid r' \in \zeta_{\uparrow,\downarrow}(r)\} \\
& \cup \{\text{ObjectSomeValuesFrom}(r,C') \mid C' \in \zeta_{\uparrow,\downarrow}(C)\} \\
\zeta_{\uparrow,\downarrow}(\text{ObjectHasSelf}(r)) = {} & \uparrow(\text{ObjectHasSelf}(r)) \\
& \cup \{\text{ObjectHasSelf}(r') \mid r' \in \zeta_{\uparrow,\downarrow}(r)\} \\
\zeta_{\uparrow,\downarrow}(\text{ObjectHasValue}(r,a)) = {} & \uparrow(\text{ObjectHasValue}(r,a)) \\
& \cup \{\text{ObjectHasValue}(r',a) \mid r' \in \zeta_{\uparrow,\downarrow}(r)\} \\
& \cup \{\text{ObjectSomeValuesFrom}(r,A) \mid A \in \zeta_{\uparrow,\downarrow}(\{a\})\} \\
\zeta_{\uparrow,\downarrow}(\text{ObjectMaxCardinality}(n,r,C)) = {} & \uparrow(\text{ObjectMaxCardinality}(n,r,C)) \\
& \cup \{\text{ObjectMaxCardinality}(n',r,C) \mid n' \in \uparrow(n)\} \\
& \cup \{\text{ObjectMaxCardinality}(n,r',C) \mid r' \in \zeta_{\downarrow,\uparrow}(r)\} \\
& \cup \{\text{ObjectMaxCardinality}(n,r,C') \mid C' \in \zeta_{\downarrow,\uparrow}(C)\} \\
\zeta_{\uparrow,\downarrow}(\text{ObjectMinCardinality}(n,r,C)) = {} & \uparrow(\text{ObjectMinCardinality}(n,r,C)) \\
& \cup \{\text{ObjectMinCardinality}(n',r,C) \mid n' \in \downarrow(n)\} \\
& \cup \{\text{ObjectMinCardinality}(n,r',C) \mid r' \in \zeta_{\uparrow,\downarrow}(r)\} \\
& \cup \{\text{ObjectMinCardinality}(n,r,C') \mid C' \in \zeta_{\uparrow,\downarrow}(C)\} \\
\zeta_{\uparrow,\downarrow}(\text{ObjectExactCardinality}(n,r,C)) = {} & \uparrow(\text{ObjectExactCardinality}(n,r,C)) \\
& \cup \{\phi_1 \sqcap \phi_2 \mid \phi_1 \in \zeta_{\uparrow,\downarrow}(\text{ObjectMaxCardinality}(n,r,C)) \\
& \quad \wedge \phi_2 \in \zeta_{\uparrow,\downarrow}(\text{ObjectMinCardinality}(n,r,C))\} \\
\zeta_{\uparrow,\downarrow}(\text{ObjectOneOf}(a_1,\dots,a_n)) = {} & \uparrow(\text{ObjectOneOf}(a_1,\dots,a_n))
\end{aligned}
$$

Using this abstract refinement operator, we build two concrete refinement operators, a generalization operator $\gamma_{\mathcal{O}} = \zeta_{\text{UpCover}_{\mathcal{O}},\text{DownCover}_{\text{O}}}$ and a specialization operator $\rho_{\mathcal{O}} = \zeta_{\text{DownCover}_{\mathcal{O}},\text{UpCover}_{\mathcal{O}}}$. Using these generalization and specialization operators we, then define the axiom weakening operator $g_{\mathcal{O}}$ for OWL 2 DL axioms as follows:

$$
\begin{aligned}
g_{\mathcal{O}}(\text{SubClassOf}(C,D)) = {} & \{\text{SubClassOf}(C',D) \mid C' \in \rho_{\mathcal{O}}(C)\} \\
& \cup \{\text{SubClassOf}(C,D') \mid D' \in \gamma_{\mathcal{O}}(D)\} \\
g_{\mathcal{O}}(\text{ClassAssertion}(C,a)) = {} & \{\text{ClassAssertion}(C',a) \mid C' \in \gamma_{\mathcal{O}}(C)\} \\
g_{\mathcal{O}}(\text{ObjectPropertyAssertion}(r,a)) = {} & \{\text{ObjectPropertyAssertion}(r',a) \mid r' \in \gamma_{\mathcal{O}}(r)\} \\
& \cup \{\text{ObjectPropertyAssertion}(r,a), \bot \sqsubseteq \top\} \\
g_{\mathcal{O}}(\text{NegativeObjectPropertyAssertion}(r,a)) = {} & \{\text{NegativeObjectPropertyAssertion}(r',a) \mid r' \in \rho_{\mathcal{O}}(r)\} \\
& \cup \{\text{NegativeObjectPropertyAssertion}(r,a), \bot \sqsubseteq \top\} \\
g_{\mathcal{O}}(\text{SameIndividual}(a_1,\dots,a_n)) = {} & \bigcup_{i=1}^{n} \{\text{SameIndividual}(\{a_1,\dots,a_n\} \setminus \{a_i\})\} \\
& \cup \{\text{SameIndividual}(a_1,\dots,a_n)\} \\
g_{\mathcal{O}}(\text{DifferentIndividuals}(a_1,\dots,a_n)) = {} & \bigcup_{i=1}^{n} \{\text{DifferentIndividuals}(\{a_1,\dots,a_n\} \setminus \{a_i\})\}
\end{aligned}
$$

$$
\begin{aligned}
&\cup \{\text{DifferentIndividuals}(a_1, \ldots, a_n)\}\\
g_{\mathcal{O}}(\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{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}}(C_i)\}\\
g_{\mathcal{O}}(\text{DisjointUnion}(D, C_1, \ldots, C_n)) &= \{\text{DisjointUnion}(D, C_1, \ldots, C_n), \bot \sqsubseteq \top\}\\
g_{\mathcal{O}}(\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{InverseObjectProperties}(s, r)) &= \{\text{InverseObjectProperties}(s, r), \bot \sqsubseteq \top\}\\
g_{\mathcal{O}}(\text{FunctionalObjectProperty}(r)) &= \{\text{FunctionalObjectProperty}(r), \bot \sqsubseteq \top\}\\
g_{\mathcal{O}}(\text{InverseFunctionalObjectProperty}(r)) &= \{\text{InverseFunctionalObjectPro}(r), \bot \sqsubseteq \top\}\\
g_{\mathcal{O}}(\text{SymmetricObjectProperty}(r)) &= \{\text{SymmetricObjectProperty}(r), \bot \sqsubseteq \top\}\\
g_{\mathcal{O}}(\text{AsymmetricObjectProperty}(r)) &= \{\text{AsymmetricObjectProperty}(r), \bot \sqsubseteq \top\}\\
g_{\mathcal{O}}(\text{TransitiveObjectProperty}(r)) &= \{\text{TransitiveObjectProperty}(r), \bot \sqsubseteq \top\}\\
g_{\mathcal{O}}(\text{ReflexiveObjectProperty}(r)) &= \{\text{ReflexiveObjectProperty}(r), \bot \sqsubseteq \top\}\\
g_{\mathcal{O}}(\text{IrreflexiveObjectProperty}(r)) &= \{\text{IrreflexiveObjectProperty}(r), \bot \sqsubseteq \top\}\\
g_{\mathcal{O}}(\text{ObjectPropertyDomain}(r, C)) &= \{\text{ObjectPropertyDomain}(r, C') \mid C' \in \gamma_{\mathcal{O}}(C)\}\\
g_{\mathcal{O}}(\text{ObjectPropertyRange}(r, C)) &= \{\text{ObjectPropertyRange}(r, C') \mid C' \in \gamma_{\mathcal{O}}(C)\}\\
g_{\mathcal{O}}(\text{SubObjectPropertyOf}(s, r)) &= \{\text{SubObjectPropertyOf}(s', r) \mid s' \in \rho_{\mathcal{O}}(s)\}\\
&\cup \{\text{SubObjectPropertyOf}(s, r') \mid r' \in \gamma_{\mathcal{O}}(r) \wedge r \text{ is simple}\}\\
&\cup \{\bot \sqsubseteq \top\}\\
g_{\mathcal{O}}(\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}}(s_i)\}\\
&\cup \{\text{SubObjectPropertyOf}(\\
&\quad \text{ObjectPropertyChain}(s_1, \ldots, s_n), r)\}\\
g_{\mathcal{O}}(\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}}(r_i)\}
\end{aligned}
$$