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 thesis compares the amount of information that the repair is able to retain relative to other repairs based on the inferred concept hierarchy. Finally, the implementation of the presented axiom weakening operator in the popular ontology editor, Protégé, is discussed.

Contents

Acknowledgements

Thanks Mami and Tati!

Chapter 1

Introduction

Chapter 2

Background and Related Work

2.1 The $\mathcal{SROIQ}$ Description Logic

Formally, an ontology is a set of statements expressed in a suitable logical language and with the purpose of describing a specific domain of interest. 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[1] 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.

In this section will briefly introduce the description logic $\mathcal{SROIQ}$ [15, 30, 3]. A more detailed description, containing more information about the semantics, can be found in Appendix A.

The Syntax of $\mathcal{SROIQ}$

The Semantics of $\mathcal{SROIQ}$

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 description 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.

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

[1]While description logics are fragments of FOL in the sense that for every ontology 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.

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, concept inclusion axioms 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) [11, 26], which is a World Wide Web Consortium (W3C) recommendation and is extensively used 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 that trade-off expressive power for more efficient reasoning [27, 26]. OWL 2 EL is a subset based on the $\mathcal{EL}^{++}$ description logic, 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 and 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 description logics like $\mathcal{SROIQ}$, it provides many more axiom and concept expession 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, datatypes and data properties. For the rest of this thesis, these will not be considered and they have been removed from the ontolgies used in the evaluation, as will be expalined in further detail in Chapter 5.

In addition to the different profiles, there exist also different syntax for OWL 2. The syntax defined and used by the main specification [26] is the so-called functional syntax. An alternative ontology format supporting OWL 2 is the XML/RDF based format [5, 25], 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 RDF format is not designed to be presented and manipulated directly by human users, but is suitable for machine processing and interoperability. The RDF format can be used also for other forms of data and can be written also in an alternative non-XML syntax that is more human-readable called Turtle [6]. Another popular format for OWL 2 is the Manchester syntax [14] which is also the default used in the popular ontology development tool Protégé (see further).

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 on the more restricted profiles, e.g., ELK [20] supports only the

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

OWL 2 EL profile. Others, like the one used for this thesis have complete or nearly 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 by means of the OWL API [12]. 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 a number of basic transformations. Further, it provides a common way of interfacing with different reasoners to query consistency, entailment, and class hierarchies.

Protégé [28] is a popular tool for ontology development build 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 used 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.

2.3 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.

2.3.1 Categories of Bugs

Defects, in both software systems and ontologies, can be due to a number of different reasons. In [17] 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. 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 $C \sqcup \sqcap D$, $\leq 3\ t.\mathit{Self}$, and $\mathit{Self} \sqcup D$ do all not match the grammar for concept expressions.

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. In the example of $\mathcal{SROIQ}$, these include the restricted use of non-simple roles and the regularity condition placed on the RBox. For programming languages, a similar restriction to this may be the requirement for definition before use or the presence of a type system[3]. These restrictions reduce the space of valid

[3]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).

programs. Restrictions of this kind can often be much easier to violate and harder to debug than the first kind of syntactic defects that is a violation of the grammar.

Example 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 simply be verified by showing that no preorder $\preceq$ exists for which $a \preceq b$ and $a \not\preceq b$ hold.

Similarly, the ontology $\mathcal{O} = \{a \circ a \sqsubseteq a, \top \sqsubseteq \exists a.\mathit{Self}\}$ is not valid 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 [17] 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. 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 $\mathcal{I}$, $a^{\mathcal{I}}$ must be in $D^{\mathcal{I}}$, but this means that $a^{\mathcal{I}}$ must not be in $C^{\mathcal{I}}$ which contradicts with the last axiom.

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. In this thesis the focus will be mainly on these kinds of bugs, especially on the problem of inconsistent ontologies.

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 or programmers intention. This can be done both for software systems and for ontologies.

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 on part of the ontology developer. These can range from trivial typos, that will most often simply result in syntactic errors, or from 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 as the result of an oversight by the modeller. One can for example define a role as transitive, and then afterwards use it in cardinality constraints. Unfortunately, however, this is not allowed in many ontology formalisms, 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 source for bugs in ontologies may arise while 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. OWL even includes an axiom that can be used to import other ontologies. This can again be a case where one ontology defines a certain role as transitive, while the other assumes that it is simple. Combining the two ontologies will then obviously lead to a violation of the global constraints. In general, the combination of different ontologies can also change which profile the ontologies fall into. Also, different ontologies might make different but incompatible modelling choices when modelling the same domain. This can result is an ontology after merging that is inconsistent or has unsatisfiable concepts.

Example 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 [4]. 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 1. 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 [4], generally an optimal repair does not necessary need to exist.

Example 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, [31, 17, 19, 2]). 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. 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 $\mathcal{O}^{(1)}, \mathcal{O}^{(2)}, \ldots$ 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 [29].

Definition 3. 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 $J_1, \ldots, J_n$ 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 $i = 1, \ldots, n$. H is a *minimal hitting* set if it does not strictly contain another hitting set.

Example 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 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 $J_1, \ldots, J_n$ 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 [18, 31, 32] 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 [18], which conveniently can compute both all justifications and all hitting sets. There exist also glass-box approach to computing justifications [18], 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 [16] or a progression-based algorithm [24] may be used. A subset of all such sets can be found efficiently using the MergeXplain algorithm [33]. Of course, computing all minimal correction subsets directly is also possible, using similar algorithms to the ones used for computing all justifications [23].

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 in order to find a repair.

Example 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 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., [10, 9, 4, 34, 8, 13, 21]).

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 leads to less information loos relative to the unmodified ontology. In [13] 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 8. Given the inconsistent ontology $\mathcal{O} = \{A \sqsubseteq B \sqcup \neg C, B \sqsubseteq C, A(a)\}$, 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* [34, 8, 7, 4, 21]. 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 4. An axiom α is *weaker* that another axioms α' with respect to some ontology $\mathcal{O}$ if and only if for every model $\mathcal{I} \models \mathcal{O}$ of $\mathcal{O}$, $\mathcal{I} \models \alpha$ implies $\mathcal{I} \models \alpha'$. Equivalently we write $\alpha \models_{\mathcal{O}} \alpha'$.

In [21] 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 [4] 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. [34] presents the repair of inconsistent ontologies using axiom weakening with the help of refinement operators. It is further empirically shown in [34] that weakening axioms can retain significantly more information compared to removing them. This approach is later extended in [8, 7] to cover more expressive description logics, including most concepts of $\mathcal{SROIQ}$. Additionally, in [8] 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 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 [34, 8] 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 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 [34]. To provide the necessary context for the following discussion on how weakeing may be performed in $\mathcal{SROIQ}$, we provide a brief overview of axiom weakening in the case of $\mathcal{ALC}$. For further details see [34]. We begin by giving the definition for 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 5. Let $\mathcal{O}$ be an $\mathcal{ALC}$ or $\mathcal{SROIQ}$ ontology. The set of *subconcepts* of $\mathcal{O}$ is given by

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

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

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

Let us now define the so-called upward and downward cover sets for concepts. The upward cover for a specific 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 [1].)

Definition 6. Let $\mathcal{O}$ be an $\mathcal{ALC}$ ontologies that using the vocabulary N_C, N_R, and N_I. The *upward cover* and *downward cover* for a concept C are given by

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

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 9. Let $N_C = \{A, B, C\}$, $N_R = \{r, s\}$, and $\mathcal{O} = \{A \sqsubseteq B, r \sqsubseteq s\}$. $\mathsf{sub}(\mathcal{O}) = \{\top, \bot, A, B\}$. The upward cover of $C \sqcup A$ is equal to $\mathsf{UpCover}_{\mathcal{O}}(C \sqcup A) = \{\top\}$. The potentiality refinement to $C \sqcup B$ will be missed even by iterated application of the upward cover because $C \sqcup B \notin \mathsf{sub}(\mathcal{O})$. Similarly, $\mathsf{UpCover}_{\mathcal{O}}(\forall r.A) = \{\top\}$, even if $\forall r.B$ and $\forall s.A$ are reasonable generalizations.

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 7. 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 by induction on the structure of concept expressions as follows.

$$\begin{aligned}
\zeta_{\uparrow,\downarrow}(A) &= \uparrow(A) \quad , A \in N_C \cup \{\top, \bot\} \ , \\
\zeta_{\uparrow,\downarrow}(\neg C) &= \uparrow(\neg C) \cup \{\neg C' \mid C' \in \zeta_{\downarrow,\uparrow}(C)\} \ ,
\end{aligned}$$

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

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

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

If we again revisit the example in Example 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 [34]. 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 not 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 8. 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

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

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

Example 10.

In order to apply the weakening operator to the problem of repairing inconsistent ontologies, we follow the algorithm depicted in Algorithm 1. First, as mention 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 [34] 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., [22]). 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 weakening.

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 $g_{\mathcal{O}^{\text{ref}}}(\alpha)$. Like for the choice of reference ontology, there are also

Algorithm 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 $\mathcal{O}$
```

multiple options for the selection of bad axioms. [34] proposes two different implementations. One based on choosing axioms at random, and another implementation that randomly samples a number of (or all the) minimal inconsistent subsets of axioms $J_1, J_2, \ldots, J_k \subseteq \mathcal{O}$ and return one axiom in $\mathcal{O}$ from the ones occurring the most often. The weaker axiom is chosen form 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 [8], that when selecting among the weakening at random using a uniform distribution, the algorithm is almost surely going to terminate. A similar effect of almost sure termination could also be achieved by maintaining a constant (small but positive) probability of removing the axiom at each step.

Example 11.

Chapter 3

Theoretical Foundations

3.1 Problems of Expressivity

Example 12. Take as an example the ontology $\mathcal{O} = \{a \circ b \circ a \sqsubseteq c\}$ that defines the simple roles a and b, and a non-simple role c. Adding the axiom $c \sqsubseteq b$, even if it is correct in isolation, will result in an ontology that is not regular.

Example 13. As another example, take the ontology $\mathcal{O} = \{a \circ a \sqsubseteq a, \top \sqsubseteq \exists c.\mathit{Self}\}$ that defines the transitive role a and simple role c used in a self-assertion. Adding the axiom $a \sqsubseteq c$, even if it may be correct in isolation, will result in a violation of the restrictions because c will become non-simple and non-simple roles may not be used in self-assertions.

3.2 RBox weakening

Let us now consider the weakening of RBox axioms, starting with the weakening of role hierarchies. Weakening role hierarchies in $\mathcal{SROIQ}$ becomes complicated due to global restrictions that are placed on the ontology. Adding an axiom to a valid $\mathcal{SROIQ}$ ontology, even if the axiom on its one may be valid, does not guarantee that the resulting ontology is still valid. Both the restrictions in the usage of simple roles and the regularity constraint on the RBox need to be considered if we want to ensure that a resulting ontology adheres to the restrictions in $\mathcal{SROIQ}$.

3.2.1 Weakening role hierarchies in $\mathcal{ALCH}$

To avoid these complications, we will first consider the simple case of weakening role hierarchies in $\mathcal{ALCH}$. $\mathcal{ALCH}$ supports only simple RIAs, and does not have any of the restrictions that are present in $\mathcal{SROIQ}$. Also, we can note that

Example 14.

3.2.2 Weakening role hierarchies in $\mathcal{SROIQ}$

Example 15.

Example 16.

Example 17.

3.3 Axiom Weakening in $\mathcal{SROIQ}$

Chapter 4

Implementation

4.1 Implementing $\mathcal{SROIQ}$ Weakening

4.2 Axiom Weakening in Protégé

Chapter 5

Experiments and Evaluation

5.1 Evaluating the Quality of Repairs

5.2 Effectiveness of Caching in Cover Computation

5.3 Evaluating Required Execution Time

Chapter 6

Conclusion and Outlook

Bibliography

[1] 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.

[2] 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.

[3] Franz Baader et al. *An Introduction to Description Logic*. Cambridge University Press, 2017.

[4] Franz Baader et al. “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.

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

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

[7] Roberto Confalonieri et al. “Irresistible Refinement Operators for Expressive Description Logics”. unpublished. 2022.

[8] Roberto Confalonieri et al. “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.

[9] Roberto Confalonieri et al. “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.

[10] 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.

[11] "OWL 2 Web Ontology Language Primer (Second Edition)". In: (2012). Ed. by Pascal Hitzler et al. URL: `https://www.w3.org/TR/owl2-primer/`.

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

[13] 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.

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

[15] 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.

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

[17] Aditya Kalyanpur et al. "Debugging unsatisfiable classes in OWL ontologies". In: *Web Semantics: Science, Services and Agents on the World Wide Web* 3.4 (2005), pp. 268–293.

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

[19] Aditya Kalyanpur et al. "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.

[20] 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.

[21] 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.

[22] Domenico Lembo et al. "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.

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

[24] 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.

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

[26] "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: `http://www.w3.org/TR/owl2-syntax/`.

[27] "OWL 2 Web Ontology Language Profiles (Second Edition)". In: (2012). Ed. by Boris Motik et al. URL: `https://www.w3.org/TR/owl2-profiles/`.

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

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

[30] 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.

[31] 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.

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

[33] 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.

[34] Nicolas Troquard et al. "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.

Appendix A

The $\mathcal{SROIQ}$ Description Logic

A.1 $\mathcal{SROIQ}$ Syntax

The *vocabulary*[1] $\mathrm{N} = \mathrm{N}_I \cup \mathrm{N}_C \cup \mathrm{N}_R$ of a $\mathcal{SROIQ}$ ontology 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 $\mathcal{SROIQ}$ 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 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^{--}, which are not valid in $\mathcal{SROIQ}$, we define a function *inv* such that $inv(r) = r^-$ and $inv(r^-) = r$. We denote the set of all roles as $\mathcal{L}(N_R) = \mathrm{N}_R \cup \{U\} \cup \{r^- \mid r \in \mathrm{N}_R\}$.

A *role inclusion axiom* (RIA) is a statement of the form $R_1 \circ \cdots \circ R_n \sqsubseteq R$ where $R, R_1, \ldots R_n \in \mathcal{L}(N_R)$ 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 $\mathcal{R}_h$.

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

[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.

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 role inclusion $S \sqsubseteq R$ where S is non-simple is itself non-simple, and
- if a role R is non-simple, then $inv(R)$ is also non-simple.

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

Example 18.

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 $S \prec R \iff inv(S) \prec R$ 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:

- $inv(R) \sqsubseteq R$,
- $R \circ R \sqsubseteq R$,
- $R \circ S_1 \circ \cdots \circ S_n \sqsubseteq R$,
- $S_1 \circ \cdots \circ S_n \circ R \sqsubseteq R$, 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 role inclusion axioms that include role chains. These types of cyclic definition could otherwise lead to undecidability of the logic.

Example 19.

Example 20.

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 [15] in that it always allows simple roles on the left-hand side similar to what has been described in [30]. However, it is more permissive than stated in [30] in that it allows for inverse roles on the right-hand side. This definition of regularity aligns with implementation of the OWL 2 DL [26] profile checker in the OWL API [12].

The set of *role assertions* $\mathcal{R}_a$ is a finite set of statements with the form $disjoint(S_1, S_2)$ (*disjointness*) where S_1, and S_2 are simple roles in $\mathcal{R}_h$. In [15] 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 $\{inv(R) \sqsubseteq R\}$, $\{disjoint(R, inv(R))\}$, $\{R \circ R \sqsubseteq R\}$, $\{r' \sqsubseteq R, \top \sqsubseteq \exists r'.Self\}$, and $\{\top \sqsubseteq \neg\exists R.Self\}$ 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].

[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'.

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 $C \in \mathrm{N}_C$ are concept, called *atomic concepts*,
- single individual names $\{i\}$ with $i \in \mathrm{N}_I$ 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 $R \in \mathcal{L}(N_R)$ 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 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.

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

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 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 $\{1\}^{\mathcal{I}} = \{i^{\mathcal{I}}\}$.
- $\neg C$ 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 $(\forall R.C)^{\mathcal{I}} = \{\delta_1 \in \Delta^{\mathcal{I}} \mid \forall \delta_2 \in \Delta^{\mathcal{I}} .\ \langle \delta_1, \delta_2 \rangle \in R^{\mathcal{I}} \rightarrow \delta_2 \in C^{\mathcal{I}}\}$.
- $\exists R.\mathit{Self}$ indicates all individuals that the extension of R connects to themselves, hence we let $(\exists R.\mathit{Self})^{\mathcal{I}} = \{\delta \in \Delta^{\mathcal{I}} \mid \langle \delta, \delta \rangle \in R^{\mathcal{I}}\}$.
- $\leq n\ R.C$ represents those individuals that have at most n other individuals they are R-related to in the concept extension of C, that is $(\leq n\ R.C)^{\mathcal{I}} = \{\delta_1 \in \Delta^{\mathcal{I}} \mid \big|\{\delta_2 \in \Delta^{\mathcal{I}} \mid \langle \delta_1, \delta_2 \rangle \in R^{\mathcal{I}} \wedge \delta_2 \in C^{\mathcal{I}}\}\big| \leq n\ \}$ where $|S|$ denotes the cardinality of a set S.
- $\geq n\ R.C$ corollary to the case above indicates those domain elements that have at least N such R-related elements,
$(\geq n\ R.C)^{\mathcal{I}} = \{\delta_1 \in \Delta^{\mathcal{I}} \mid \big|\{\delta_2 \in \Delta^{\mathcal{I}} \mid \langle \delta_1, \delta_2 \rangle \in R^{\mathcal{I}} \wedge \delta_2 \in C^{\mathcal{I}}\}\big| \geq n\ \}$.

Example 21.

A.2.2 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 disjointness axiom $disjoint(S, R)$ hold iff the extensions of R and S are disjoint, formally $\mathcal{I} \models disjoint(S, R) \iff S^{\mathrm{I}} \cap R^{\mathrm{I}} =$.
- 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 ontology $\mathcal{O}$, written $\mathcal{I} \models \mathcal{O}$, iff $\mathcal{I}$ satisfies every axiom $\alpha \in \mathcal{O}$ of the ontology, i.e., $\mathcal{I} \models \mathcal{O} \iff \forall \alpha \in \mathcal{O} \,.\, \mathcal{I} \models \alpha$. If $\mathcal{I}$ satisfies $\mathcal{O}$, we say $\mathcal{I}$ is a model of $\mathcal{O}$.

A.2.3 Reasoning tasks

In general, logic-based knowledge representation is useful for the ability to perform reasoning task on ontologys. 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 *ontology satisfiability, axiom entailment, and concept satisfiability.*

Knowledge base satisfiability

A knowledge $\mathcal{O}$ base is satisfiable iff there exists a model $\mathcal{I} \models \mathcal{O}$ for $\mathcal{O}$. Otherwise, the ontology is called unsatisfiable, inconsistent, or contradictory. 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 unsatisfiable 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 $\mathcal{I} \models \mathcal{O}$ of the ontology also satisfies the axiom $\mathcal{I} \models \alpha$. 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

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 $\neg R(a_1, a_{n+1})$
$disjoint(S, R)$	$S(a, b)$ 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. This means, checking entailment of α with respect to $\mathcal{O}$, is equivalent to checking unsatisfiability of $\mathcal{O} \cup B$. a, a_i, and b are assumed to not appear in $\mathcal{O}$.

are entailed by $\mathcal{O}$, we write $\text{Con}(\mathcal{O}) = \{\alpha \mid \mathcal{O} \models \alpha\}$. 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 for 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 $\mathcal{I} \models \alpha \iff \mathcal{I} \not\models \beta$. 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 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.

Concept satisfiability

A concept C is satisfiable with respect to $\mathcal{O}$ iff there exists a model of the ontology $\mathcal{I} \models \mathcal{O}$ 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 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 $\mathcal{I} \models \mathcal{O}$ 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 satisfies $\mathcal{I} \models C \sqsubseteq \bot$, meaning $\mathcal{O} \models C \sqsubseteq \bot$. We conclude that we can test for 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 22. OWL has an axiom $\mathrm{DisjointClasses}(C_1, \dots, C_n)$[1] that allows specifying that a set of classes all are pairwise disjoint. $C_i \sqcap C_j \sqsubseteq \bot$ for all $i \neq j = 1, \dots, n$. 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 23. The OWL axiom $\mathrm{EquivalentClasses}(C_1, \dots, C_n)$ 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, $\mathsf{UpCover}_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}$ and $\mathsf{DownCover}_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}$, are used. We define the refinement operator $\zeta_{\uparrow,\downarrow}$ for OWL 2 DL as follows:

$$\begin{aligned}
\zeta_{\uparrow,\downarrow}(A) &= \uparrow(A) \qquad \text{for } A \in \mathrm{N_c} \cup \mathcal{L}(N_R) \cup \{\top, \bot\} \\
\zeta_{\uparrow,\downarrow}(\mathrm{ObjectComplementOf}(C)) &= \uparrow(\mathrm{ObjectComplementOf}(C))
\end{aligned}$$

[1]In the rest of this section, the OWL 2 functional syntax (cf. [26]) will be used.

$$\begin{aligned}
&& \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{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{ObjectOneOf}(a_1,\dots,a_n)) &= & {\uparrow}&(\text{ObjectOneOf}(a_1,\dots,a_n))\\
\text{OWL 2 concepts:}\quad &&&\\
\zeta_{\uparrow,\downarrow}(\text{ObjectExactCardinality}(n,r,C)) &= & {\uparrow}&(\text{ObjectExactCardinality}(n,r,C))\\
&& \cup\,&\{\alpha_1 \sqcap \alpha_2 \mid \alpha_1 \in \zeta_{\uparrow,\downarrow}(\text{ObjectMaxCardinality}(n,r,C))\\
&& &\quad \wedge \alpha_2 \in \zeta_{\uparrow,\downarrow}(\text{ObjectMinCardinality}(n,r,C))\}\\
\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\})\}
\end{aligned}$$

Using this abstract refinement operator, we build two concrete refinement operators, a generalization operator $\gamma_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}} = \zeta_{\mathsf{UpCover}_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}},\mathsf{DownCover}_{\mathcal{O}}}$ and a specialization operator $\rho_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}} = \zeta_{\mathsf{DownCover}_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}},\mathsf{UpCover}_{\mathcal{O}^{\text{ref}},\mathcal{O}^{\text{full}}}}$. 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 axioms, but stands in for some tautological axiom that is true in every possible interpretations.

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

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

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{InverseFunctionalObjectPro}(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 between OWL 2 DL and $\mathcal{SROIQ}$

Appendix D

User Guide: Java Axiom Weakening Library

Appendix E

User Guide: Protégé Plugin