Thesis Title

Roland Bernard

July 2023

Abstract

Contents

Chapter 1

Introduction

Chapter 2

Background

2.1 Software Bugs vs 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 defects

Defects, in both software systems and ontologies, can be due to a number of different reasons. In [3] 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.

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. These restrictions reduce the space of valid 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.

Semantic defects

For ontologies, semantic defects, as defined in [3] 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.

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 Debugging Defects

2.2 Ontologies and 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 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) [5, 1], 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 [4]. The OWL 2 DL profile is the most expressive of the profiles, while still being decidable, and is based on $\mathcal{SROIQ}$.

The following sections will introduce the description logic $\mathcal{SROIQ}$ [2] and the OWL 2 language. The relation between the two will also be discussed. The description is also based on the description in [6].

2.2.1 The $\mathcal{SROIQ}$ Description Logic

2.2.2 $\mathcal{SROIQ}$ Syntax

The *vocabulary* $\mathrm{N} = \mathrm{N}_I \cup \mathrm{N}_C \cup \mathrm{N}_R$ 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. The set of role names contains the *universal role* $u \in \mathrm{N}_R$.

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* of the form r or r^- for some role names $r \in \mathrm{N}_R$, where r^- is called the *inverse role*. For convenience in the latter definitions, and to avoid roles like r^{--}, we define a function Inv such that $\mathrm{Inv}(r) = r^-$ and $\mathrm{Inv}(r^-) = r$. We denote the set of all roles as $\mathbf{R} = \mathrm{N}_R \cup \{\, r^- \mid r \in \mathrm{N}_R \,\}$.

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. A finite set of RIAs is called a *role hierarchy*, denoted $\mathcal{R}_h$.

There is an additional restriction that is placed upon the role hierarchy in a $SROIQ$ knowledge base. The role hierarchy in $SROIQ$ 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 $s \prec r \iff s^- \prec r$ for all roles names 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 $r \in \mathrm{N}_R$ is a (non-inverse) role name and $s_i \prec r$ 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.

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 $\mathbf{R}^S = \mathbf{R} \setminus \mathbf{R}^N$.

The set of *role assertions* $\mathcal{R}_a$ is a finite set of statements with the form $\mathrm{Ref}(r)$ (*reflexivity*), $\mathrm{Asy}(s)$ (*asymmetry*), or $\mathrm{Dis}(s_1, s_2)$ (*disjointness*), where r is a (possibly non-simple) role and s, s_1, and s_2 are simple roles in $\mathcal{R}_h$. In [2] the authors define additionally the role assertions $\mathrm{Sym}(r)$ (*symmetry*), $\mathrm{Tra}(r)$ (*transitivity*), and $\mathrm{Irr}(r)$ (*irreflexivity*). These additional assertions can, however, be written using the alternative axioms $r^- \sqsubseteq r$, $r \circ r \sqsubseteq r$, and $\top \sqsubseteq \neg\exists r.\mathrm{Self}$ respectively.

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
- $a = b$ (*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.

2.2.3 $\mathcal{SROIQ}$ 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 $\mathcal{I} \vDash \alpha$. 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 $\mathcal{I} \vDash s_1 \circ \cdots \circ s_n \sqsubseteq r \iff s_1^{\mathcal{I}} \circ \cdots \circ s_n^{\mathcal{I}} \subseteq r^{\mathcal{I}}$ 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, $\mathcal{I} \vDash \mathrm{Ref}(r) \iff \{ \langle \delta, \delta \rangle \mid \delta \in \Delta^{\mathcal{I}} \} \subseteq r^{\mathcal{I}}$.
- A role asymmetry axioms $\mathrm{Asy}(r)$ is holds $\mathcal{I} \vDash \mathrm{Asy}(r)$ 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 $\mathcal{I} \vDash \mathrm{Dis}(s, r) \iff s^{\mathrm{I}} \cap r^{\mathrm{I}} =$.
- A general concept inclusion axiom $C \sqsubseteq D$ is true iff the extension of C is fully contained in the extension of D, hence $\mathcal{I} \vDash C \sqsubseteq D \iff C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$.
- 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 $\mathcal{I} \vDash C(a) \iff a^{\mathcal{I}} \in C^{\mathcal{I}}$.
- 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 $\mathcal{I} \vDash r(a, b) \iff \langle a^{\mathcal{I}}, b^{\mathcal{I}} \rangle \in r^{\mathcal{I}}$.
- A negative role assertion $\neg r(a, b)$ is true exactly than when the corresponding role assertion $r(a, b)$ is false. Equivalently, $\mathcal{I} \vDash \neg r(a, b) \iff \langle a^{\mathcal{I}}, b^{\mathcal{I}} \rangle \notin r^{\mathcal{I}}$.
- An equality assertion $a = b$ holds iff the individuals identified by a and b are the same element of the domain, alternatively written $\mathcal{I} \vDash a = b \iff a^{\mathcal{I}} = b^{\mathcal{I}}$.
- Dual to the above, $a \neq b$ holds iff the names a and b denote different elements, accordingly $\mathcal{I} \vDash a \neq b \iff a^{\mathcal{I}} \neq b^{\mathcal{I}}$.

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 $\mathcal{I} \vDash \mathcal{KB}$, iff $\mathcal{I}$ satisfies every axiom $\alpha \in \mathcal{KB}$ of the knowledgeable, i.e., $\mathcal{I} \vDash \mathcal{KB} \iff \forall \alpha \in \mathcal{KB} \,.\, \mathcal{I} \vDash \alpha$. 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 $\mathcal{I} \vDash \mathcal{KB}$ for $\mathcal{KB}$. Otherwise, the knowledge base is called unsatisfiable, inconsistent, or contradictory. As discussed in section 2.1.1, 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 $\mathcal{I} \vDash \mathcal{KB}$ of the knowledge base also satisfies the axiom $\mathcal{I} \vDash \alpha$. We also write this as $\mathcal{KB} \vDash \alpha$ 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.

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 $\mathcal{I} \vDash \alpha \iff \mathcal{I} \nvDash \beta$. 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. Table 2.1 shows the correspondence for every type of $\mathcal{SROIQ}$ axiom.

α	B
$s_1 \circ \cdots \circ s_n \sqsubseteq r$	$s_1(a_1, a_2)$, …, $s_n(a_n, a_{n+1})$, and $\neg r(a_1, a_{n+1})$
$\mathrm{Asy}(r)$	$r(a, b)$ and $r(b, a)$
$\mathrm{Ref}(r)$	$\neg r(a, a)$
$\mathrm{Dis}(s, r)$	$s(a, b)$ 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 $\mathcal{KB} \cup B$.

Concept satisfiability A concept C is satisfiable with respect to $\mathcal{KB}$ iff there exists a model of the knowledge base $\mathcal{I} \vDash \mathcal{KB}$ 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 $\mathcal{I} \vDash \mathcal{KB}$ 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 satisfies $\mathcal{I} \vDash C \sqsubseteq \bot$, meaning $\mathcal{KB} \vDash C \sqsubseteq \bot$. We conclude that we can test for unsatisfiability of a concept C by checking for entailment of $C \sqsubseteq \bot$.

2.2.4 The Web Ontology Language

2.3 Ontology Repair

2.4 Axiom Weakening

Chapter 3

??

Bibliography

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

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

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

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

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

[6] 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.