University of Zurich^UZH

Department of Informatics

End User Assisted Ontology Evolution in Uncertain Domains

Doctoral Thesis
for the Degree of
Doctor of Informatics

Faculty of Economics,
Business Administration and
Information Technology of the
University of Zurich

by Thomas Scharrenbach
from Andernach, Germany

Accepted on the recommendation of

Prof. Abraham Bernstein, PhD
Prof. Dr. Ulrike Sattler
Dr. Bettina Waldvogel
PD Dr. Rolf Grütter

2012

The Faculty of Economics, Business Administration and Information Technology of the University of Zurich herewith permits the publication of the aforementioned dissertation without expressing any opinion on the views contained therein.

Zurich, March 2012

The Vice-Dean of the Academic Program in Informatics:
Prof. Dr. Harald Gall

Acknowledgements

First of all I wish to thank my supervisor Avi Bernstein who gave me the possibility to write this thesis. Thanks, Avi for your support, your countless ideas and integrating me, the external student, into your group. Most of all, thanks for your never-ending optimism and motivation during all the ups and downs.

Thanks also to Uli Sattler who agreed to take the burden of being my external examiner. Never would I have thought to appreciate having such a pedant corrector. I would like to thank you, Uli for all your critical comments. Those gave me an even deeper insight into the guts of Description Logics.

My heartfelt thanks go also to Bettina Waldvogel, my ”motherly” supervisor at WSL. Thank you, Bettina, for all your support and all your patience. Furthermore, cheers to all the Waldvogels, who gave us such a warm welcome to Switzerland and did everything to give us a good start in a new country.

I would further like to thank my other ”pedant” supervisor Rolf Grütter for countless discussions. Thank you, Rolf, for your critical and valuable comments and the various inputs you gave me, in particular for geo-spatial reasoning.

Special thanks go to my office mates Kathi, Jonas and Adrian. Thank you for all the coffee, cookies, the fun and the serious discussions. For all the feedback you gave me on papers and to this thesis. And of course for cheering up in hard times.

For all the fun and laughter and all those serious real-life discussions, I would like to thank my colleagues at WSL. I had a great time with you lot at Birmensdorf.

Thanks to my former and current colleagues at DDIS. Although only there once a week, you made me really feel being part of the group. Thank you for your support, all those arguments at the whiteboard and the feedback you gave me–especially you, Juk.

For all the nice and fruitful discussions I'd like to say thank you to Matthew Horridge and Pavel Klinov from Manchester. And of course for all the fun we had at the conferences.

Thanks to Claudia d'Amato and Nicola Fanizzi for the kind invitation to Bari. I had a great time there with you and the rest of the Bari team.

Thanks go also to the bunch of anonymous reviewers for their effort reviewing the papers I submitted some of which were accepted for publication, some of which were not. I have a deep respect for your work and appreciate your critical and constructive comments which helped me improving this work each time.

All those friends that have been there or came along the way. I appreciate your friendship, Heidi, Olaf, Nils und Clarissa, Norbert und Tine, Arne, André and Evangelia. You will always have a place in my heart—as well as all those that got lost on the way and those that I forgot to mention.

Thanks go also to the members of the Förderverein für die Freizeitstätte Kelberg e. V. Especially the inner circle around Hans, Michael, Elmar and Wolfgang for always putting me back to the ground and for being ready to have a beer or two at any time.

For their endless support I would like to thank my Mum and Dad. Thank you for always being my backup. Thanks also to my parents-in-law and the rest of the family.

Finally, I would like to thank the most important person in my life. Daniela, without your love, your support and patience writing this thesis would have been impossible. Thank you for being there whenever I needed you.

Abstract

In the beginning of the Semantic Web, ontologies were usually constructed once by a single knowledge engineer and then used as a static conceptualization of some domain. Nowadays, knowledge bases are increasingly dynamically evolving and incorporate new knowledge from different heterogeneous domains — some of which is even contributed by casual users (i.e., non-knowledge engineers) or even software agents. Given that ontologies are based on the rather strict formalism of Description Logics and their inference procedures, conflicts are likely to occur during ontology evolution. Conflicts, in turn, may cause an ontological knowledge base to become inconsistent and making reasoning impossible. Hence, every formalism for ontology evolution should provide a mechanism for resolving conflicts.

Ontology Repair is a way to solve contradictions. Yet, it is a complex task that is usually performed by experts in knowledge engineering. It requires a deep understanding of the underlying logical formalism to be able to understand what consequences a repair on an ontology implies. Hence, Ontology Repair is very time-consuming.

In this thesis we provide with an intermediate solution that allows drawing logical consequences in the presence of contradictions. We use procedures of Lehmann's Default Logics and Lukasiewicz' Probabilistic Description Logics putting axioms that cause contradictions in the TBox into different drawers. By disentangling the root causes for contradictions we weaken the information present but achieve absence of contradictions —one major reason for inconsistencies in knowledge bases that have a Description Logic as underlying formalism for drawing logical consequences. While we have to accept that potentially interesting logical consequences are invalidated, the approach presented in this thesis allows us to keep all explicitly stated axioms, i.e. there is no need to remove information that a user has provided.

We provide a general framework for solving contradictions in Description Logics knowledge bases without the need for changing the knowledge representation. We define our approach as well as classical full Ontology Repair in terms of this operator to make both approaches comparable. Experiment on example ontologies show that our approach can outperform full Ontology Repair in terms of invalidated logical consequences.

Zusammenfassung

Als das Semantische Web noch in den Kinderschuhen steckte, wurden Ontologien üblicherweise als Einzelarbeit eines Experten in Wissensmodellierung erstellt und danach als unveränderliche Konzeptualisierung einer Domäne betrachtet. Im Laufe der Zeit wurden aber aus solch statischen Wissensbasen mehr und mehr als dynamische und somit veränderliche Systeme, insbesondere durch hinzufügen von Information aus anderen, heterogenen Domänen. Neues Wissen wird mehr und mehr durch Laien hinzugefügt oder sogar durch maschinelle Agenten.

Da Ontologien auf Beschreibungslogiken - also formallogischen Systemen zum logischen Schlussfolgern - aufbauen, schleichen sich häufig Fehler in Form von Widersprüchen ein, wenn Nicht-Experten in Wissensmodellierung einer Ontologie neues Wissen hinzufügen. Solche Widersprüche wiederum können dazu führen, dass eine Wissensbasis logisch inkonsistent wird und somit den Prozess der logischen Schlussfolgerung ad absurdum führt. Somit sollte jedweder Formalismus, der die Evolution von Ontologien beschreibt Methoden zum Umgang mit Widersprüchen aufweisen.

Das Reparieren von Ontologien ist eine oft gewählte Möglichkeit, um Widersprüche aufzulösen. Problematisch dabei ist, dass es dazu Experten in Wissensmodellierung braucht, welche ein tiefes Verständnis für die zugrunde liegenden Formalismen

aufweisen. Sie müssten zudem in der Lage sein, zu verstehen, welche logischen Konsequenzen eine Reparatur nach sich zieht. Aus diesem Grund ist das Reparieren von Ontologien ein zeitaufwändiger Prozess.

In dieser Arbeit wird eine Zwischenlösung erarbeitet, welche logisches Schlussfolgern trotz Widersprüchen in der Ontologie erlaubt. Angelehnt an Methoden von Lehmann's Default-Logiken und Lukasiewicz' Probabilistischen Beschreibungslogiken werden die Axiome der TBox in verschiedene Schubladen gesteckt. Somit werden die eigentlichen Ursachen für die Widersprüche auseinanderdividiert. Wenn auch die Aussagekraft der vorhandenen Axiome geschwächt wird, so werden Widersprüche in jedem Fall aufgelöst und somit wiederum eine wichtige Ursache für Inkonsistenzen auf Wissensbasen, deren Semantik auf Beschreibungslogiken basieren, beseitigt. Auf der einen Seite werden damit auch potentiell interessante implizite Schlussfolgerungen beseitigt; auf der anderen Seite erlaubt der in dieser Arbeit erarbeitete Ansatz, alle explizit spezifizierten Axiome zu behalten. Mit anderen Worten: Sämtliche Information, welche explizit durch einen Benutzer hinzugefügt wurde, bleibt stets erhalten.

In dieser Arbeit wird ein genereller Operator zum Auflösen von Widersprüchen in Wissensbasen, welche auf Beschreibungslogiken basieren, vorgestellt. Dieser lässt den Formalismus zur Wissensrepräsentation unverändert und erlaubt den direkten Vergleich des Reparierens von Ontologien mit dem im Rahmen dieser Dissertation erarbeiteten Ansatz, wozu beide Verfahren mittels des Operators definiert werden. Experimente mit Beispiel-Ontologien zeigen, dass der vorgestellte Ansatz beim Auflösen von Widersprüchen potentiell weniger Schlussfolgerungen beseitigt als der Reparatur-Ansatz.

Table of Contents

Part I

Introduction and Preliminaries

1 Introduction

Contradiction is not a sign of falsity,
nor the lack of contradiction a sign of truth

Blaise Pascal (1623-1662)
French mathematician and philosopher

Knowledge in the Semantic Web is represented by ontologies expressed in the Web Ontology Language OWL. The current standard, OWL 2 [W3C OWL Working Group, 2009], defines different profiles that, in turn, allow to different levels of expressivity. All of these profiles have some Description Logics as a rough syntactic variant. Description Logics (DL) are decidable fragments of first-order logics (FOL) where explicit knowledge is expressed in axioms and assertions.[1] DL knowledge bases have well-defined model-theoretic semantics that allows drawing new conclusions from existing knowledge.

Evolution may lead to conflicts

When ontologies evolve, knowledge is removed or new knowledge is added. By adding new axioms and/or new assertions, contradictions may be introduced that cause the knowledge base as a whole to be inconsistent. For example, adding axioms that express disjointness the ontology may infer some

[1] There indeed exist undecidable DL but these are usually not used within the scope of the Semantic Web.

concept U unsatisfiable. When there exists an assertion of some individual or instance u to that concept, the whole knowledge base is inferred inconsistent. Yet, for an inconsistent knowledge base any conclusion—even meaningless ones—becomes trivially true. It is hence desirable to prevent concepts from being inferred unsatisfiable. There indeed exist more reasons for why a knowledge base can become inconsistent, but *we propose to start off with conflict-free conceptualizations and present a method that never infers any concept to be unsatisfiable*.

Hide the Complexity

Domain experts vs Knowledge Engineers

A central challenge for ontology evolution is that there can be two parties involved: Knowledge engineers, on the one hand, are trained to construct conflict-free knowledge bases, but they usually lack domain knowledge. On the other hand, there are the domain experts who have the knowledge that shall be formalized by an ontology. Unfortunately, they are usually not experienced knowledge engineers. Studies in cognitive psychology, like done by Ceraso and Provitera [1971], found out that human agents make systematic errors while formulating or interpreting logical descriptions. Similar results were observed for the task of developing of OWL-ontologies, too [Rector et al., 2004; Doran et al., 2009]. This makes us assume that the following proposition is true:

Proposition 1. *Evolving OWL-ontologies are very likely to be erroneous.*

In the scope of this thesis we consider the case that OWL-ontologies are developed by domain experts, i.e. non-experts in knowledge engineering. Knowledge engineers are highly skilled specialists. Yet, they lack domain knowledge, and it is not always possible to get such an expert for developing an OWL-ontology. We hence assume the following Proposition 2 to be true in the scope of this thesis:

Proposition 2. *OWL-ontologies are developed by non-experts in knowledge engineering.*

Ontologies can be developed in an explicit or in an implicit way. In the first case, agents directly add statements to an ontology. This can be done by editing the XML-serialization of an ontology, but there exist a variety of specialized editors for building ontologies like Protégé,[2] or the TopBraid-Composer.[3] A detailed overview over different ontology editors can be found on Wikipedia.[4]

In the second case, ontologies transparently integrate into a system. Examples are social software systems like (semantic) wikis or social networks. When, for example, users add content to a semantic wiki [Krötzsch et al., 2007], statements are added to the ontology without noticing it, and consequently, without being able to take control over the implied logical consequences.

Editing ontologies requires expertise.

Letting non-experts in knowledge-engineering develop an OWL-ontology explicitly bears the risk of severe problems. The development of OWL-ontologies is already a hard task for them. Since removing conflicts requires a more detailed understanding of the underlying Description Logics, *non-experts in knowledge engineering are expected to introduce logical conflicts but they cannot be expected to be able to solve them.* To avoid possible problems with the acceptance of OWL-ontologies as a way of formalizing knowledge and/or motivation to develop such ontologies, the ontology development process must keep away the complexity of the underlying formal logics from the developing agent. We hence assume the following Proposition to hold:

Proposition 3. *If non-experts in knowledge engineering shall evolve a DL knowledge base, the complexity of the underlying logics must be transparent to them.*

[2] http://protege.stanford.edu/

[3] http://www.topbraidcomposer.com/

[4] http://en.wikipedia.org/wiki/Ontology_Editor

Transparency is not to be mixed up with hiding the complexity. If we just hide the complexity, the users will still have to live with the consequences of contradictions. In case of transparency, those conflicts and their consequences are not exposed to the users. This becomes even more important when agents implicitly evolve an ontology, i.e. when there is little or no control over the logical consequences that are introduced when interacting with the system.

Solving Conflicts

Contradictions may be caused by agents that automatically assemble new information from different heterogeneous sources. This is likely to happen when combining different ontologies, modeled by different agents. But even ontologies created by knowledge engineers may contain conflicting pieces of knowledge.

We stated in Proposition 3 the need for removing the complexity of the underlying logical formalism for the user. As a consequence, conflicts and their consequences shall not be exposed to the agents working with an ontology.

As such, we make the following proposition:

Proposition 4. *Any framework for ontology evolution has to provide a way for resolving conflicts.*

Ontology Repair

Resolving conflicts is usually referred to as *Ontology Repair*. In that sense, this thesis provides a solution for Ontology Repair, too. In order not to mix up terms, we will rather refer to ”solving conflicts”. For ontology evolution we require more than the absence of conflicts. Agents querying (or interacting with) an ontology in the Semantic Web assume that answers to queries are expressible in OWL 2. Furthermore, the answer should have meaningful semantics and should not contain conflicts.

A popular method for ontology repair is removing and/or rewriting axioms [Kalyanpur et al., 2005]. This method only partially satisfies the desired properties listed above. Indeed, the knowledge representation remains the same. Axiom rewriting, in contrast, requires manual effort. Whilst axiom removal can be done automatically, both original as well as some inferred knowledge will be lost. Methods extending the formalism for reasoning usually require changing the knowledge representation. Finally, approaches to only perform reasoning on non-conflicting parts of the knowledge base invalidate given knowledge [Huang et al., 2005].

The nature of conflicts

The notion of what is a conflict depends on the formalism used for drawing logical consequences. In the case of OWL (and consequently, DL) conflicts refer to axioms that lead to contradictions under classical Aristotelean logics: A statement and its complement cannot be true at the same time. If we require that the underlying logical formalism must not change, only removing and/or rewriting axioms can resolve such contradictions.
If we do not change the formalism for drawing logical consequences, we can resolve contradictions only by removing and/or changing explicitly stated knowledge. While preserving coherency and/or consistency, we consider the need for removing explicitly stated information as a flaw for knowledge systems to which non-experts in knowledge engineering contribute their knowledge.

Instead of removing axioms (i.e. explicit knowledge), we propose to invalidate those inferences (i.e. implicit knowledge) that cause concepts to be inferred unsatisfiable. This can have severe consequences for the underlying formalism for drawing logical consequences, one of which is the need for giving up monotonicity. As we will investigate in more detail later on, other consequences can be changing the formalism for knowledge representation, for example by changing to multi-valued logics.

The price of solving conflicts

Resolving conflicts in OWL 2 knowledge bases is a compromise between giving up certain properties of the statements in the knowledge base. As stated before, we consider removing explicitly given statements as undesirable, but accept not to be able to draw certain (implicit) logical consequences anymore. Changing the formalism for knowledge representation is considered a faux pas, because we risk agents to lose the ability to communicate with each other.[5] Hence, we require every method for solving conflicts in OWL 2 knowledge bases to fulfill the following properties:

Proposition 5. *Any formalism for ontology repair has to fulfill the following properties:*

P1 Preservation of expressive power: The formalism for knowledge representation is not changed.

P2 Coherency: No concept is inferred unsatisfiable[6].

P3 Autonomy: The procedure shall work automatically.

P4 Conservation of explicit information: The original explicitly stated information should be kept.

P5 Conservation of implicit information: As little inferred information as possible shall be lost.

We should note that Proposition 5 can neither be proved or disproved. Nevertheless, we think that these properties reflect a good compromise between rigid proper repair and disabling the ability to draw logical consequences at all. We think that the ability for drawing logical consequences in the presence of contradictions is higgley desirable but should not be too different from what classical entailment does. We will pick up this issue again in the discussion in Chapter 9.

[5] Indeed, a central idea behind OWL 2 is to have a standard format for knowledge exchange in the Semantic Web.

[6] For the current work we concentrate on resolving unsatisfiable concepts. The procedure may, in principle, be extended to resolve unsatisfiable roles and—to a certain extent—assertions.

Our Approach

We present an approach for solving contradictions for the terminological part of an OWL 2 knowledge base. Solving the contradictions satisfies property P2. Furthermore, we keep all original statements while having to remove certain (implicit) logical consequences. We hence fulfill property P4.

We show that we lose fewer inferences than in the case of axiom removal meeting the requirements of property P5. Finally, we are able to keep the formalism for knowledge representation when invalidating inferences but not axioms. Namely, we consider DL knowledge bases so that the first desired property P1 is satisfied, and apply the following reasoning approach.

Solving conflicts by separating axioms

Based on Lehmann's Default Logics [Lehmann, 1995] and Probabilistic Description Logics [Lukasiewicz, 2008] our approach separates axioms responsible for a conflict into different partitions and collects axioms not involved in a conflict in its separate set. Due to the separate treatment of conflict-causing axioms, inferences for unsatisfiable concepts are invalidated but the formalism for knowledge representation remains unchanged. To limit the number of inferences lost when reasoning the single partitions are considered alongside with the set of non-conflict axioms. We call the pair of the partition and the set of non-conflicting axioms a *Multi-View TBox*. We introduce the Δ-operator that maps any TBox $\mathcal{T}$ to a Multi-View TBox $\mathcal{DT}$ such that $\mathcal{DT}$ contains exactly those axioms of the original TBox $\mathcal{T}$.

While this method, unfortunately, cannot prevent that some potentially interesting inferences are lost, it limits the loss to a minimum. It can, furthermore, be shown that potentially fewer inferences are lost than in the case of full axiom removal—while still working fully automatically and hence meeting requirement P3

Any formalism for ontology evolution must strictly fulfill the *hard properties* P1, P2 and P3, whereas we require the *soft properties* P4 and P5 only to be satisfied as much as possible. Since axiom removal is the only formalism known to fulfill the strict properties—besides the procedure proposed in this work—we use it as the baseline for comparing it to our approach w.r.t. the soft properties.

Optimal Procedures

Optimal procedures are non-deterministic

As the case of ontology repair by axiom removal, the investigated approach is non-deterministic. In addition to that, the resulting Multi-View TBox can still contain certain types of conflicts.[7] Although these conflicts do not cause any (meaningless) inference to become valid—as it can be the case using classical DL reasoning—they can provide confusing information. It is hence desirable to reduce these to a minimum. *However, solutions that minimize the number of inferences invalidated do not necessarily minimize the number of conflicts.* Finding optimal solutions hence requires an evaluation measure that assesses solutions, i.e. Multi-View TBoxes not only by the size of their deductive closure. Instead, minimizing conflicts requires a more qualitative measure.

We show that our approach outperforms Ontology Repair by the number of lost inferences. Yet, finding optimal solutions for either task requires to compute the causes or justifications. Hence both procedures, our approach as well as axiom removal, rely upon a task that is 2NEXPTIME-complete. Fortunately, efficient procedures for finding justifications is a subject that became more and more popular over the last years, and it could be shown that it indeed works for many real-world OWL-ontologies.

Applications

This thesis was motivated by work done in the project Data Center Nature and Landscape (DNL) [Frehner and Brändli, 2006] at

[7] As required by the coherency property, still no concept is inferred unsatisfiable

the Swiss Federal Institute for Forest, Snow and Landscape Research (WSL).[8] Different heterogeneous distributed data sources should be made commonly accessible. Since most of the data was collected by or on the behalf of public authorities in Switzerland, the problem turned out to be even multi-lingual. Ontologies were designed to allow non-expert users an open and intuitive multi-lingual search on the various data sources [Bauer-Messmer and Grütter, 2007]. One of the challenges was the lack of knowledge engineers, which lead to the idea of letting domain experts that created and/or work with the data regularly create and/or evolve the ontologies [Bauer-Messmer et al., 2008]. The rationale was that they know best what the data is actually about. We experienced in the scope of the DNL project that domain experts are not very enthusiastic about getting into the details of formal logics. An observation we also made when letting people from the street evolving a small-scale DL knowledge base at a public science event ”Nacht der Forschung 2009”.[9]

A motivating use case

We see the main application for this approach in the context of ontology evolution. Especially when human agents are involved who are not experts in knowledge engineering.[10] Using the proposed approach, they can add their information to the knowledge base and still be able to infer meaningful results—including their very own information. Our approach hence provides a pseudo-coherent intermediate state for the knowledge base that undergoes proper repair by a knowledge engineer from time to time. Since we compute the root causes for the contradictions, we provide additional support for this task. While arguable, we think that the properties of Proposition 5 provide the

Use cases that may benefit from our approach.

[8] http://www.wsl.ch

[9] http://www.nachtderforschung.ethz.ch/

[10] Which does, in turn, not mean that only knowledge engineers are capable of evolving an OWL ontology in a consistent way. We just assume that the risk of introducing errors is lower for a person with a background in knowledge engineering than for a person who has not.

best compromise between correctness (i.e. performing proper repairs) and usability.

Knowledge engineers who evolve ontologies can benefit from our approach, too. Particularly, when the evolution takes place in a distributed environment, and agreement on the actual repair can only be achieved from time to time.

While repair is one aspect, there exist also situations where it is indeed desirable to be able to keep contradictory pieces of information in the knowledge base. In particular, when exceptional information is to be modeled but the formalism for knowledge representation does not directly allow for it — which is the case when using OWL.

Another area where we can apply our methods is Ontology Matching. In that case, two ontologies are mapped onto each other. This can easily lead to conflicts, because the two conceptualizations which underly the ontologies can be contradictory. Yet, a repair can be undesirable, when, for example, the present axioms must not be changed. Furthermore, if a repair is desired, it may not immediately be available. This can, as stated above, happen when no knowledge engineer is available. Applying our approach, it is hence possible to draw conclusions while the ontologies together with the mapping may be incoherent.

To sum up, our approach can be useful, whenever ontologies have to remain incoherent for a while before they can be repaired.

1.1 Contribution and Hypotheses

This work makes two contributions:

1. We show how Lehmann's Default Logics can be used for consistent ontology evolution without the need of removing axioms.

2. We introduce an entropy-based measure for finding optimal solutions, since finding solutions is a non-deterministic process.

The first is the main contribution of this work, whereas the latter refers to optimization. We give a formal proof for the main contribution and show how the approach can be further extended.

The usefulness of these contributions can be summarized in the following working hypotheses:

Hypothesis 1. *Using our approach fewer inferences are lost than in the case of axiom removal.*

Hypothesis 2. *Minimizing the number of conflicts does not result in maximizing the number of invalidated inferences.*

Hypothesis 3. *The entropy-based measure minimizes the number of conflicts and the number of invalidated inferences.*

We do not provide a formal proof but provide an empirical evaluation in Chapter 7. We will see from the results that Hypotheses 1 and 2 can be accepted whereas we have to reject Hypothesis 3.

1.2 Referenced Own Publications

Works of the author

First ideas about consistent ontology evolution were published at the *22nd International Conference on Environmental Informatics (EnviroInfo 2008)* [Bauer-Messmer et al., 2008]. The initial proposal for this thesis was published at the *Doctoral Consortium of the 7th International Semantic Web Conference 2008 (ISWC2008)* [Scharrenbach, 2008].

The process of how to avoid unsatisfiable concepts for ontology evolution was described at the *1st Workshop on Inductive Reasoning and Machine Learning for the Semantic Web (IRMLeS2009)* co-located with the *European Semantic Web Conference 2009 (ESWC2009)* [Scharrenbach and Bernstein, 2009].

The actual partitioning scheme was presented at the *23rd International Workshop on Description Logics 2010 (DL2010)* [Scharrenbach et al., 2010c]. A refined version of this partitioning scheme was presented at the *4th International Workshop on Ontology Dynamics (IWOD2010)* [Scharrenbach et al., 2010b] whereas the principles of the optimization using TBox entropy was published at the *6th International Workshop on Uncertainty Reasoning for the Semantic Web (URSW 2010)* [Scharrenbach et al., 2010a]. Both events were co-located with the *9th International Semantic Web Conference 2010 (ISWC2010)*.

1.3 Plan of this Thesis

This work is divided into five parts: After a motivation we give an introduction to the fundamental concepts in Part I. In the subsequent Part II we describe the methods used in this thesis which are evaluated in Part III. We reflect the work alongside with its relation to other work in Part IV. For better readability we put long proofs in the Appendix or Part V.

Part I: Introduction

In particular, the remainder of the first part provides an overview over OWL, Description Logics and justifications in Chapter 2. In that chapter we also introduce the basic notion of Default Logics.

Part II: Methods

The methods part starts with basic notions and the fundamentals of our approach towards invalidating inferences in Chapter 3. There we introduce the general Δ-operator for invalidating inferences and show how classical ontology repair as well as our methods can be formalized in term of a Δ-operator. Using these formal structures for the Δ-operator as a basis, we show how Default Logics, in particular, can be used for solving conflicts. The core of our procedure is the so-called Unsat-Splitting which we introduce in Chapter 4. We there provide certain restrictions on the TBox needed for consistent solutions and introduce some Lemmas that will be important for the consistency proofs later on. Finally, each Unsat-Splitting-Δ-Operator is illustrated by an algorithmic definition in Chapter 5.

Part III: Evaluation

The non-determinism of the Unsat-Splitting-Δ-operators introduced before requires optimization techniques for finding optimal solutions. Hence, the evaluation starts with formalizing optimization techniques in Chapter 6. An empirical evaluation of the Δ-operators and the optimization techniques follows in Chapter 7.

Part IV: Reflection

In the reflection we discuss the findings of this work in Chapter 9, and give an overview over related work in Chapter 8. In the final Chapter 10 we conclude the thesis and give an outlook on future work.

Part V: Appendix

The Appendix starts with a glossary for the most commonly used terms and symbols in Appendix A. Detailed proofs that were not presented in the text can be found in Appendix B.

2

Preliminaries

Ontologies in the Semantic Web are expressed in the Web Ontology Language OWL.[1] The current standard, OWL 2 [Hitzler et al., 2009] defines several profiles, all of which refer to a certain Description Logic (DL) or a fragment thereof. In turn, DLs are (decidable) fragments of first-order logics. Evolving an OWL 2 ontology effectively means evolving a first-order logics knowledge base.[2] Especially when non-experts in knowledge engineering perform this task they potentially introduce contradictions to the knowledge base. Resolving those contradictions usually involves finding their causes or justifications.

Plan of This Chapter

In the following, we will give a short introduction to the fundamental concepts that are used throughout the thesis. We give an overview over Description Logics and its relation to OWL and

[1] We do not want to neglect that there exist more possibilities to define and formalize ontologies. In the scope of this thesis we restrict out view on OWL-ontologies.

[2] In the scope of this thesis we do not distinguish between ontology evolution and ontology design.

first-order logics in Section 2.1. In Section 2.2 we provide details about lattice theory and its application to certain properties on DL-knowledge bases. Afterwards we show in Section 2.3 how justifications are defined in the context of DL and how to compute them. Subject to Section 2.4 is that the notion of contradictions depends on the underlying logical system. There we investigate how different logical systems may be used to avoid contradictions. Finally, Default Logics are described in Section 2.5. In particular, the variant using Lehmann's Lexicographical Entailment as inference procedure is presented, as it serves as the basis for the approach of conflict solving that is investigated in this work.

2.1 Description Logics

Logic is logic. That's all I say.

Oliver Wendell Holmes (1809-1894),
American Physicist and Writer,
The One-Hoss Shay

Description Logics (DL) are languages for knowledge representation with a well-defined syntax and semantics. A DL allows to specify *concepts* (also known as classes), instances thereof, also referred to as *individuals* and binary relations, called *roles* between individuals. Operators such as *negation* ($\neg$) or *conjunction* ($\sqcap$) enable the construction of complex concepts and roles.

Concepts are used to represent abstract or general knowledge about the world. Consider, for example, the following complex concept that represents things that have at most two wings:

$$\leq 2 hasWing.Wing$$

Statements about concepts are called axioms. They allow to express simple if-$then$ rules:

$$Bird \sqsubseteq \leq 2hasWing.Wing$$

The intended meaning of this axiom is the following: If we learn that something is a bird, then it must not have more than two wings. In terms of DL we thereby require all instances that are asserted to the concept of $Bird$ to $have$ at most two things that are known to be instances of the concept of $Wing$.

In the example above we also used the role $hasWing$ to express that something can have wings. Indeed, the complex concept $hasWing.Wing$ restricts the range of this role: We require that the role ranges only over instances of the concept $Wing$. We will see in this section, how we can give an expressive description of the world by using concepts, roles, instances and axioms.

DLs are based on formal logic-based semantics and hence also provide methods how to implicitly infer new knowledge from the given knowledge. We refer to this as *reasoning* or *inference*. Yet, inference must not be mixed up with *entailments* or *logical consequences*. The latter are those pieces of knowledge that follow from the formal logics whereas inferences are those pieces of knowledge that the procedures we define on the logics find.

Reasoning and Inference

When using DLs to formalize and reason about knowledge, we make a so-called *Open-World Assumption* (OWA) [Reiter, 1987a]. We cannot falsify something that cannot explicitly be inferred false. In the example above we can entail that an instance from which we know that it has more than two wings cannot be a bird. Yet, we have no evidence for this inference to be true, in case we know that this instance only had one wing. Even if we knew that something has exactly one wing, we would not be able to infer that it is not a bird—we would have to also state that birds must have at least two wings. In that sense, DL lacks so-called *negation-as-failure* [Clark, 1987], because if we cannot prove a statement to be true, we implicitly assume that it is false. In the following we will give a formal introduction to the basics of Description Logics. For further information, the interested reader is referred to Baader and Nutt [2007].

2.1.1 Basic Description Logics

DLs are distinguished by the set of language constructors they provide. These, in turn, define the *expressive power* of a DL.[3] Elementary descriptions are called atomic concepts and atomic roles. Using constructors of a DL language we can inductively build complex descriptions. We usually denote atomic concepts by A, B and atomic roles by R whereas complex concepts are usually denoted by C, D and complex roles by S.

The basic DL language is called $\mathcal{AL}$ and provides the following constructors for building complex concepts:

$$\begin{array}{lcrcl}
C, D & \rightarrow & A & | & \text{atomic concept} \\
 & & \top & | & \text{top concept} \\
 & & \bot & | & \text{bottom concept} \\
 & & \neg A & | & \text{atomic negation} \\
 & & C \sqcap D & | & \text{intersection} \\
 & & \forall R.C & | & \text{value restriction} \\
 & & \exists R.\top & & \text{limited existential quantification}
\end{array}$$

where A is an atomic concept, C and D are (possibly complex) concept descriptions and R is a role name. Extensions to the basic language are union of concepts ($\mathcal{U}$), full existential quantification ($\mathcal{E}$), number or cardinality restrictions in unqualified ($\mathcal{N}$) as well as qualified form ($\mathcal{Q}$), and the complement or negation of arbitrary complex concepts ($\mathcal{C}$). Concepts may also be defined in terms of a closed set of instances (which will be defined later on in this section) in the form of nominals ($\mathcal{O}$).

[3] We refer to a DL language simply to as DL, where the meaning is clear from the context.

C, D	$\rightarrow$	$C \sqcup D$	$\mid$	concept union
		$\exists R.C$	$\mid$	full existential quantification
		$\geq nR.\top$	$\mid$	at-least (cardinality) restriction
		$\leq nR.\top$	$\mid$	at-most (cardinality) restriction
		$\geq nR.C$	$\mid$	qualified at-least (cardinality) restriction
		$\leq nR.C$	$\mid$	qualified at-most (cardinality) restriction
		$\neg C$	$\mid$	complement/negation of complex concepts
		$\{a, b, c\}$		nominals, i.e. closed sets of instances

where now a, b, c are names of concept instances.

We may also apply operators to the roles to increase the expressive power. A role may be restricted to be functional ($\mathcal{F}$). Further, we can allow for a role-hierarchy ($\mathcal{H}$), transitive roles ($\mathcal{S}$), inverse roles ($\mathcal{I}$), limited complex role inclusion axioms,[4] as well as reflexivity, irreflexivity($\mathcal{R}$).

Usually, a datatype theory $\mathcal{D}$ over concrete domains [Schmidt-Schauss and Smolka, 1991; Lutz and Miličić, 2007] (like the natural numbers, reals, finite strings etc.) is provided for roles, i.e. the range of a role may be restricted to certain datatypes such as intervals of integers, strings matching a regular expression etc. The currently most expressive Web Ontology Language, OWL 2, has the DL $\mathcal{SROIQ}(\mathcal{D})$ [Horrocks et al., 2006] as a rough syntactic variant. More information on the naming scheme for Description Logics can be found in Schmidt-Schauss and Smolka [1991]; Baader and Nutt [2007].

We abbreviate axioms and assertions by lower-case Greek letters as $\tau : D \sqsubseteq C$, $\rho : S \sqsubseteq R$, and $\alpha : C(d)$.

[4] We will specify the term ”axiom” below.

2.1.2 DL Ontologies and DL Knowledge Bases

Terminologies

A DL allows to define a *terminology* of concepts. Such a terminology, called a *TBox*, is a finite set of *axioms* describing a *hierarchy* on a set of concepts. TBox axioms are of the form $Bird \sqsubseteq Animal$ and express, for example, that the concept $Bird$ is a specialization (or subsumed by) of the concept $Animal$. We also refer to the concept $Bird$ as the left-hand-side or sub-concept of this *subclass-inclusion axiom*, whereas we refer to $Animal$ as the right-hand-side or the super-concept. The semantics of subclass-inclusion axioms expresses the constraint that each instance of the concept of $Bird$ also to be an instance of the concept $Animal$.

Depending on its expressive power, a DL may define role inclusion axioms and role assertions. The first can be used to define a *role hierarchy* and/or complex roles while the latter is used to describe properties of roles like symmetry or transitivity. A finite set of role inclusion axioms and role assertions is called an RBox denoted by $\mathcal{R}$.

Assertions about individuals are expressed as, for example, $Bird(crow\_127)$, $livesIn(peter, switzerland)$ and $red \neq blue$ and are referred to as *concept assertions*, *role assertions*, and *inequality assertions*, respectively. A finite collection of assertions about individuals is called an *ABox*.

We define a DL *ontology* $\mathcal{O}$ as a pair of a TBox $\mathcal{T}$ and an RBox $\mathcal{R}$: $\mathcal{O} = \langle \mathcal{T}, \mathcal{R} \rangle$. A DL knowledge base is defined by the pair $\mathcal{K} = \langle \mathcal{O}, \mathcal{A} \rangle$ where $\mathcal{O} = \langle \mathcal{T}, \mathcal{R} \rangle$ is an ontology and $\mathcal{A}$ is an ABox—each of which is finite and possibly empty. The sets of names of concepts, roles and instances must be mutually disjoint. It is, for example, not allowed that an instance is a concept or a role at the same time.

Signatures of axioms

The signature identifies entities. The signature of a TBox axiom τ, denoted by $sig(\tau)$ is the set of all atomic concepts that occur in this axiom. Analogously we define the signature for RBox axioms as the set of all simple roles in the axiom and the signature for an ABox assertion as the set of all individuals occurring in the assertion. The signature of an TBox, RBox and ABox are defined as the union of the signatures over the corresponding axioms and assertions. The signatures for an ontology and a knowledge base are defined analogously.

In the context of Default Logics, which will be introduced in Section 2.5, we have to make statements about the left-hand-side and the right-hand-side of an axiom. For expressive knowledge bases, these are not atomic but rather complex concepts, and similarly, complex roles. We may, for example, have an axiom $A \sqcap B \sqsubseteq C \sqcap D$. For this axiom both, the left-hand-side $A \sqcap B$ as well as the right-hand-side $C \sqcap D$ are complex concepts. In the scope of this work we are therefore not only interested in the atomic concepts of a TBox, but also in the complex concepts. In particular, we would like to be able to reference those complex concepts that form the left-hand-side and/or the right-hand-side of an axiom. We hence define the *extended signature* of an axiom:

Definition 1 (Extended Signature)**.** *Let τ be an axiom. The extended signature of $\tau = D \sqsubseteq C$ is defined as $\widetilde{sig}(\tau) = \{C, D\}$ where C and D may be complex concepts.*

Example 1. *Consider, for example, the axiom $\tau = Bird \sqsubseteq Animal \sqcap \exists hasWings.\top$. For this axiom we have that $sig(\tau) = \{Bird, Animal\}$. It only consists of atomic axioms. For the extended signature, on the other hand, we have that $\widetilde{sig}(\tau) = \{Bird, Animal, \exists hasWings.\top, Animal \sqcap \exists hasWings.\top\}$.*

We define the extended signature of a TBox, respective RBox, as the union of the extended signature of all its axioms. In the following we omit the term extended and simply refer to the signature of a TBox, respective RBox.

Expressive Power of a Description Logic

Not only requires Property P1 leaving the formalism for knowledge representation unchanged. We must also be able to express the same concepts, roles and axioms when evolving an ontology. The ontology that is subject to evolution is defined according to a certain OWL 2-profile. If we want to be able to express the same axioms we not only require to be able to express the new ontology according to OWL 2, but also according to that very OWL 2 profile. As OWL 2-profiles have a DL as a rough syntactic variant, *we require the underlying DL to remain unchanged when evolving an ontology*. We therefore have to be able to identify that DL.

Defining expressive power

Given an ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{R} \rangle$ we would like to be able to know of which expressive power (also called expressivity) [Baader, 1996] the underlying DL is.[5] As could be seen before, each letter of a DL language refers to a certain set of operators. If we, for example, add the concept union operator to the basic language $\mathcal{AL}$ we obtain the language $\mathcal{ALC}$. Obviously, we can express any axiom of $\mathcal{AL}$ also in $\mathcal{ALC}$.

In order to be able to identify a particular Description Logic $\mathcal{L}$, we denote the set of all possible Description Logics by $\mathbb{DL}$. For a certain DL $\mathcal{L}$ we want to be able to check whether an ontology can be expressed in terms of $\mathcal{L}$:

Definition 2 (Expressive Power)**.** *We say that a TBox $\mathcal{T}$, respective an RBox $\mathcal{R}$ can be expressed by a DL $\mathcal{L}$, if all elements of $\mathcal{T}$, respective $\mathcal{R}$ can be constructed according to the language constructors that $\mathcal{L}$ defines. If $\mathcal{T}$ and $\mathcal{R}$ can be expressed by $\mathcal{L}$, then we say that the ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{R} \rangle$ can be expressed by $\mathcal{L}$.*
(Adopted from Baader [1996])

Ordering DLs by Expressive Power

Not every ontology in a certain DL can be expressed in another DL. There exist more expressive and less expressive DLs. We can hence define a partial order $\leq_{\mathbb{DL}}$ on $\mathbb{DL}$.

[5] We can reformulate this as looking for the minimal expressive power of a TBox and an RBox.

Definition 3 (Partial Order on DLs)**.** *Let $\mathcal{L}_1$ and $\mathcal{L}_2$ be two Description Logics. We say that $\mathcal{L}_1 \leq_{\mathbb{DL}} \mathcal{L}_2$, if every TBox $\mathcal{T}$ that can be expressed by $\mathcal{L}_1$ can also be expressed by $\mathcal{L}_2$. If $\mathcal{L}_1 \leq_{\mathbb{DL}} \mathcal{L}_2$ and $\mathcal{L}_2 \leq_{\mathbb{DL}} \mathcal{L}_1$, then we say that $\mathcal{L}_1$ and $\mathcal{L}_2$ are of the same expressive power. If neither $\mathcal{L}_1 \leq_{\mathbb{DL}} \mathcal{L}_2$ nor $\mathcal{L}_2 \leq_{\mathbb{DL}} \mathcal{L}_1$, then we say that the expressive power of $\mathcal{L}_1$ and $\mathcal{L}_2$ is not comparable.*
(Adopted from Baader [1996])

When evolving an ontology the current axioms may have a less expressive power than we would allow. For example, assume we want to evolve a new, empty TBox $\mathcal{T}$ with axioms that can be constructed according to the DL $\mathcal{SRIOQ}$. If we add the axiom $C \sqsubseteq A \sqcup B$ to $\mathcal{T}$ the TBox can be expressed in $\mathcal{ALC}$ —which is of less expressive power than $\mathcal{SROIQ}$.

Leaving expressive power unchanged.

Demanding preservation of expressive power for methods for solving conflicts (P5 in Proposition 5) requires not changing the formalism for knowledge representation. We must hence ensure that the result of such a method is still able to express axioms according to the same DL as was the original ontology. The operator $\mathbb{L}$ allows to assign the *maximal expressive power* of an ontology by explicitly specifying a DL:

Definition 4 (Maximal Expressive Power)**.** *Let $\mathcal{O} = \langle \mathcal{T}, \mathcal{R} \rangle$ be an ontology. The maximal expressive power of $\mathcal{O}$ is identified by explicitly assigning $\mathcal{O}$ a DL:*

$$\mathbb{L}(\mathcal{O}) = \mathcal{L}$$

Conversely, if for an ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{R} \rangle$ the underlying DL is not specified, we implicitly assume that it is determined by a DL $\mathcal{L}$ such that $\mathcal{O}$ can be expressed by $\mathcal{L}$ and $\mathcal{L}$ is of minimal expressive power w.r.t. $\leq_{\mathbb{DL}}$.[6]

[6] If we assumed $\mathcal{L}_{max}$ to be the most expressive DL possible, then we could also define $\mathcal{L}_{max}$ as the DL that is assumed for $\mathcal{O}$ by default. The actual choice depends on the application.

2.1.3 Semantics of Description Logics

Description Logics have a well-defined semantics. Unlike in other formalisms like frame systems or semantic networks there is no ambiguity between the relationships for concept subsumption, role inclusion or assertions about individuals. In the same manner, the semantics for the language constructors and the inference process is well-defined, too.

Interpretations

The formal semantics of a DL is given by an *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$. It consists of the *interpretation domain* $\Delta^{\mathcal{I}}$, a non-empty finite set, and the *interpretation function* $\cdot^{\mathcal{I}}$ which assigns to every atomic concept A a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and to every atomic role R a set $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$.

There exist two special cases, i.e. when the interpretation of a concept is either empty or equal to the whole domain. The first case applies to the so-called *bottom-concept* denoted by $\bot$ whereas the second case applies to the so-called *top-concept*, denoted by $\top$. Since we require an interpretation to be non-empty, the interpretations for a $\bot^{\mathcal{I}} = \emptyset$ and $\top^{\mathcal{I}} \neq \emptyset$ can never be the same.

For complex concepts, the interpretation function can be extended inductively:

$$\begin{array}{rcl}
\top^{\mathcal{I}} & = & \Delta^{\mathcal{I}} \\
\bot^{\mathcal{I}} & = & \emptyset \\
(\neg A)^{\mathcal{I}} & = & \Delta \setminus A^{\mathcal{I}} \\
(C \sqcap D)^{\mathcal{I}} & = & C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
(\forall R.C)^{\mathcal{I}} & = & \{a \in \Delta^{\mathcal{I}} \,|\, \forall b.(a,b) \in R^{\mathcal{I}} \rightarrow b \in C^{\mathcal{I}}\} \\
(\exists R.\top)^{\mathcal{I}} & = & \{a \in \Delta^{\mathcal{I}} \,|\, \exists b.(a,b) \in R^{\mathcal{I}}\}
\end{array}$$

We say that two concepts C and D are equivalent, if $C^{\mathcal{I}} = D^{\mathcal{I}}$ denoted by $C \equiv D$. We say that a concept C is *satisfiable*, if its interpretation is not empty, i.e. when $C^{\mathcal{I}} \neq \emptyset$. In contrast to that, we say that C is *unsatisfiable*, if $C \equiv \bot$, i.e. $C^{\mathcal{I}} = \emptyset$. We usually denote unsatisfiable concepts by U.

Unsatisfiable concepts

Further extensions of the interpretation function for complex concepts and roles can be found in the DL Handbook [Baader and Nutt, 2007].

Satisfiability and Consistency

In order to find out whether there exist contradictions in a DL knowledge base $\mathcal{K} = \langle\langle\mathcal{T}, \mathcal{R}\rangle, \mathcal{A}\rangle$, we have to find out whether there exists an interpretation that satisfies $\mathcal{K}$. An interpretation $\mathcal{I}$ is said to *satisfy* a concept C or a role R, if the result of the interpretation function is not empty. We denote this by $\mathcal{I} \models C$, if $C^{\mathcal{I}} \neq \emptyset$ and $\mathcal{I} \models R$, if $R^{\mathcal{I}} \neq \emptyset$, respectively.

For concept subsumption and role inclusion, satisfiability is defined w.r.t. set inclusion. An interpretation $\mathcal{I}$ is said to satisfy an inclusion axiom $\mathcal{I} \models D \sqsubseteq C$, if $D^{\mathcal{I}} \subseteq C^{\mathcal{I}}$. A TBox $\mathcal{T}$ is said to be satisfiable, if there exists an interpretation $\mathcal{I}$ such that $\mathcal{I}$ satisfies every axiom in $\mathcal{T}$. In this case, we say that $\mathcal{I}$ is a *model* of $\mathcal{T}$, denoted by $\mathcal{I} \models \mathcal{T}$. If $\mathcal{T}$ has a model, then we say that $\mathcal{T}$ is satisfiable and else we say that $\mathcal{T}$ is unsatisfiable.

Satisfiability for roles w.r.t. an RBox is defined analogously. As such, we call an ontology $\mathcal{O} = \langle\mathcal{T}, \mathcal{R}\rangle$ satisfiable, if both $\mathcal{T}$ and $\mathcal{R}$ are satisfiable. In the following, we always assume the RBox to be satisfiable. Hence, an ontology is satisfiable when its TBox is satisfiable.

An ABox $\mathcal{A}$ is *consistent* with respect to an ontology $\mathcal{O} = \langle\mathcal{T}, \mathcal{R}\rangle$, if there exists an interpretation that is a model of both $\mathcal{A}$ and $\mathcal{O}$. An ABox $\mathcal{A}$ is said to be consistent, if it is consistent w.r.t. an ontology with an empty TBox and an empty RBox. A knowledge base $\mathcal{K} = \langle\langle\mathcal{T}, \mathcal{R}\rangle, \mathcal{A}\rangle$ is called *consistent*, if $\mathcal{T}$ and $\mathcal{R}$

are satisfiable and $\mathcal{A}$ is consistent w.r.t. $\mathcal{T}$. A knowledge base that is not consistent is called *inconsistent*.

If the knowledge base $\mathcal{K}$ is inconsistent, then, by definition, neither the TBox nor the RBox have a model (w.r.t. the ABox). This is particularly the case when the ABox is inconsistent w.r.t. $\mathcal{T}$ but also when no concept in $\mathcal{T}$ is satisfiable. As we will see in Section 2.1.6 DL correspond to first-order logics. Yet, a first-order logics theory that has no model implies any formula. Hence, a DL knowledge base that is inconsistent entails any axiom or assertion.

Deductive Closure

Logical consequences

An axiom $D \sqsubseteq C$ is a *logical consequence* of a TBox $\mathcal{T}$, denoted by $\mathcal{T} \models D \sqsubseteq C$, if all models of $\mathcal{T}$ satisfy $D \sqsubseteq C$. The same holds for RBoxes analogously. An assertion $C(i)$ is a logical consequence of a knowledge base $\mathcal{K} = \langle\langle\mathcal{T},\mathcal{R}\rangle,\mathcal{A},\rangle$, if $\mathcal{A} \cup \{\neg C(i)\}$ is inconsistent with $\mathcal{O}$. We denote logical consequences for a knowledge base by $\mathcal{K} \models C(i)$.

The set of all logical consequences for a TBox (resp. RBox, ABox) is called its *deductive closure* denoted by $(\mathcal{T})^\star$ (resp. $(\mathcal{R})^\star$, $(\mathcal{A})^\star$). The deductive closure of an ontology $\mathcal{O} = \langle\mathcal{T},\mathcal{R}\rangle$ is the union of the deductive closure of its components:

$$(\mathcal{O})^\star = (\mathcal{T})^\star \cup (\mathcal{R})^\star$$

The deductive closure of a knowledge base $\mathcal{K} = \langle\langle\mathcal{T},\mathcal{R}\rangle,\mathcal{A},\rangle$ is analogously defined as:

$$(\mathcal{K})^\star = (\mathcal{O})^\star \cup (\mathcal{A})^\star$$

The deductive closure for a knowledge base (resp. part thereof) is, in principle, infinite. If, for example, $D \sqsubseteq C$ is a logical consequence of $\mathcal{T}$, so are $D \sqsubseteq C \sqcap C$ and $D \sqsubseteq C \sqcap C \sqcap \ldots$. Similar statements hold for RBoxes and ABoxes.

Note that the logical consequence operator $\models$ for a knowledge base can also be expressed as function between sets of axioms and assertions (or sentences in its original version). This operator, also referred to as *Tarskian consequence operator* [Tarski, 1983b, 2002], is denoted by $Cn(\mathcal{K}) = (\mathcal{K})^\star$.

Explicit and Implicit Logical Consequences

We distinguish between explicit logical consequences and implicit logical consequences. Since a model by definition satisfies all axioms and assertions in a knowledge base $\mathcal{K}$ (resp. part thereof), any axiom of $\mathcal{K}$ is also a logical consequence for $\mathcal{K}$. We call any axiom or assertion that is contained in a part of $\mathcal{K}$ *explicit logical consequence*. In turn, we call any element of $(\mathcal{K})^\star \backslash \mathcal{K}$ *implicit logical consequence*.

Induced Logical Consequences

There exist some logical consequences that always hold for DLs. These refer to declaration and to negation. We consider those *induced logical consequences*, which are described in the following, as explicit and assume that they are contained in the TBox.

For any TBox $\mathcal{T}$ we have the declaration $\mathcal{T} \models C \sqsubseteq \top$ for any concept C in the signature of $\mathcal{T}$. We can do this, because $C^\mathcal{I} \subseteq \Delta^\mathcal{I}$ by definition. Since $\bot^\mathcal{I} = \emptyset \subseteq C^\mathcal{I}$ we know that $\mathcal{T} \models \bot \sqsubseteq C$, too. Furthermore, since $C^\mathcal{I} \subseteq C^\mathcal{I}$ we have that $\mathcal{T} \models U \sqsubseteq U$. Yet, we will only refer to concept declarations explicitly when the concept is complex and unsatisfiable. We will see later on that we will need these declarations when we want to solve unsatisfiable complex concept. We will omit declarations for atomic concepts, since they do not play a role in solving unsatisfiability.

For any concept U which is unsatisfiable w.r.t an interpretation $\mathcal{I}$ we know that $\mathcal{I} \models U \sqsubseteq \bot$. In particular, $\mathcal{T} \models U \sqsubseteq \bot$, if $\mathcal{I}$ is a model for $\mathcal{T}$. We hence identify concepts that are unsatisfiable w.r.t. a TBox $\mathcal{T}$, if $\mathcal{T} \models U \sqsubseteq \bot$. The same holds for RBoxes, if the top role and the bottom role (the role ”counterparts” to the top and bottom concept, respectively) are defined [Horrocks et al., 2006].

Explicit disjoint axioms

We consider an axiom that expresses disjointness between two concepts, i. e. $D \sqsubseteq \neg C$ and its counterpart $C \sqsubseteq \neg D$ as the same axiom. If two concepts C and D are stated disjoint in the presence of the axiom $D \sqsubseteq \neg C$, we also consider the counterpart $C \sqsubseteq \neg D$ as explicit. From this first axiom we learn that $D^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$. Assume now that there exists i such that $i \in C^{\mathcal{I}}$. We know that i cannot be included in $\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$, and hence also not in $D^{\mathcal{I}}$. As a consequence, $i \in \Delta^{\mathcal{I}} \setminus D^{\mathcal{I}}$. This means nothing else than $\mathcal{T} \models C \sqsubseteq \neg D$. The two disjoint axioms $D \sqsubseteq \neg C$ and its counterpart $C \sqsubseteq \neg D$ have the same interpretations.

2.1.4 Description Logics Reasoning

A central task for DL knowledge representation system (KRS) is to infer implicit knowledge from explicitly stated knowledge. For a DL knowledge base, a KRS defines a *reasoning problem* which usually consists of the following tasks:

1. Subsumption Test:
   - determine whether a concept D is subsumed by concept C
   - determine whether a role S is included by role R.
2. Satisfiability Test:
   - find out whether $\mathcal{T} \models U \sqsubseteq \bot$ for a concept U.
   - find out whether $\mathcal{R} \models R \sqsubseteq \bot$ for a role R.
3. Consistency Test:
   - check whether a knowledge base is consistent.

4. Instance Retrieval:

    - find those concept assertions $C(i)$ that are logical consequences for $\mathcal{K}$ for a given concept C and an individual in the signature of $\mathcal{K}$.
    - find those roles assertions $R(i, j)$ that are logical consequences for $\mathcal{K}$ for a given role R and the individuals i, j in the signature of $\mathcal{K}$.

Inconsistency is problematic

The subsumption test induces a hierarchy on the set of concepts and the set of roles. Testing for satisfiability allows the detection of possible modeling errors that may cause a knowledge base to become inconsistent. If a knowledge base is inconsistent, every possible axiom is inferred satisfiable. Hence, checking for consistency means checking whether it is possible to draw meaningful conclusions from a knowledge base. Accordingly, instance retrieval can be interpreted as the repeated application of instance checking for all known individuals of $\mathcal{K}$ and a given concept C or role R.

DL Reasoners

Soundness and completeness

A *reasoner* defines algorithms for solving the reasoning problem. The implicit axioms and assertions it produces are called inferences. Let $\mathcal{K}$ be an arbitrary DL knowledge base (or part thereof).[7] and $\mathfrak{R}$ a DL reasoner. If every inference of $\mathfrak{R}$ is a logical consequence of $\mathcal{K}$, then we say that $\mathfrak{R}$ is *sound*. If for $\mathcal{K}$ all entailments are inferences of $\mathfrak{R}$ then we say that $\mathfrak{R}$ is *complete*. Note that soundness does not imply completeness and vice versa. In the following we assume that a reasoner $\mathfrak{R}$ is sound and complete.

Most DL Reasoners are based on the tableaux calculus [Horrocks et al., 1999]. The currently most used systems are Racer [Haarslev and Möller, 1999], Pellet [Sirin et al., 2007], Fact++ [Tsarkov and Horrocks, 2005] and HermiT [Motik et al., 2009b].

[7] If we restrict reasoning, for example, to TBox reasoning

In contrast, in the DL reasoner KAON2 [Motik, 2004] reasoning is implemented reducing a $\mathcal{SHIQ}(\mathcal{D})$ knowledge base to a disjunctive datalog program [Ullman, 1988]. All of the mentioned DL-reasoners are sound and complete.

Tableau

Tableau-based reasoners try to build an abstract ABox $\mathcal{A}_i$ such that the knowledge base $\mathcal{K} = \langle \mathcal{R}, \mathcal{T}, \mathcal{A}_i \rangle$ has a model for a given RBox $\mathcal{R}$ and a given TBox $\mathcal{T}$ at the same time [Schmidt-Schauss and Smolka, 1991]. The objective is to show that the negation of a formula is not satisfiable. Therefore, all concept descriptions are transformed into negational normal form (NNF), i.e. negation occurs only in front of atomic concepts.

Tableaux are data structures, in particular trees, labeled with formulae, in our case ABox assertions. For each axiom type, *transformation rules* are defined, and those rules define the construction of an ABox while they preserve satisfiability of the formula. Since some of the axioms, for example disjunction, require non-deterministic choices, the current path in the tree is split into branches. Hence, the transformation rules actually define a set of ABoxes.

Tableaux: Proving by counter-example

The formula for which we want to proof satisfiability is indeed satisfiable, if we can find an ABox that is complete and consistent. An ABox is called complete, if none of the transformation rules applies to it, and it is called consistent, if it does not contain any *clash* (we give a more detailed explanation of ABox inconsistencies in Section 2.4.1). If one of the ABoxes is complete and does, furthermore, not contain a clash, then it can be shown that there exists a model for $\mathcal{K}$. The tableau methods can be extended such that an existing ABox can constrain the construction of the abstract ABox.

The subsumption problem turns out to be undecidable, if the DL language is of too much expressive power [Schmidt-Schauß, 1989; Patel-Schneider, 1989]. On the other hand, even for inexpressive DL languages there do not exist procedures that can decide the problem in polynomial time [Levesque and Brachman, 1987; Nebel, 1988]. Nevertheless, several optimization techniques emerged over the last years which make DL reasoning possible in practice [Horrocks and Patel-Schneider, 1999; Haarslev and Möller, 1999; Tsarkov et al., 2007; Motik et al., 2009c].

For a more detailed overview over tableau algorithms for DL reasoning, we refer to Baader and Sattler [2001].

2.1.5 Description Logics and OWL

Knowledge in the Semantic Web is represented by ontologies which are usually expressed in the *Web Ontology Language* (OWL). The current standard, OWL 2 [W3C OWL Working Group, 2009], was designed in a way such that it has the DL $\mathcal{SROIQ}$ as a rough syntactic variant [Horrocks et al., 2009]. $\mathcal{SROIQ}$ is a very expressive language, and its reasoning problems are of high computational complexity. In particular, the reasoning problem for $\mathcal{SROIQ}$ is 2 NEXPTIME-complete [Horrocks et al., 2006; Horrocks and Sattler, 2004; Kazakov, 2008].

Yet, this expressive power is not needed in every case. As a consequence, so-called profiles were defined for OWL 2, each of which correspond to a unique DL [Motik et al., 2009a]. Hence, every OWL 2 profile has well-defined semantics, and the reasoning problem is decidable. The profiles were defined in a way such that the knowledge representation is unchanged but language constructors are restricted in a way such that the corresponding DL is less expressive.

Thanks to the general nature of the method proposed in this thesis we may consider any DL for which the reasoning problem is decidable. In other words: we may apply our approach to any OWL 2 profile.

2.1.6 Description Logics and First Order Logics

Description Logics are a family of decidable fragments of First Order Logics (FOL). An interpretation $\mathcal{I}$ assigns every atomic concept an unary relation and to every role a binary relation over $\Delta^{\mathcal{I}}$. Hence, atomic concepts and roles can be considered unary and binary predicates, respectively. We can hence translate a concept C into a formula $\phi_C(x)$ with one free variable x in predicate logic such that for every interpretation $\mathcal{I}$ the set of elements from $\Delta^{\mathcal{I}}$ that satisfy $\phi_C(x)$ is exactly $C^{\mathcal{I}}$.

Expressing axioms as FOL formulas.

The FOL formula, for example, that corresponds to the complex concept $A \sqcap B$ is $\forall x : (A \wedge B)(x)$. The axiom $D \sqsubseteq C$ can be translated into the FOL rule $\forall x : D(x) \rightarrow C(x)$. We will make use of this correspondence when applying Default Logics to Description Logics in Section 2.5.

For further information about first-order logics including a historical overview over its emergence in the 20th century the interested reader is referred to Ferreiros [2001].

2.2 Posets and Lattices

Order without liberty and liberty without order are equally destructive.

Theodore Roosevelt

Graph-structures can be used to encode certain properties for sets of entities. Monotone properties, like the consequence relation of classical Aristotelean logic, define a partial order on the entities involved. Partially ordered sets—or posets—can be further restricted. Requiring that, for example, every pair of nodes must have a unique common ancestor and a unique common descendant, a poset can be considered as an algebra. Such posets are referred to as *lattices*.

Lattices for properties of TBoxes

For DL TBoxes, lattices can be used to define monotone properties on sets of axioms. Examples for this are the concept subsumption hierarchy (i.e. TBox classification) and concept unsatisfiability but also the dependency between sets of axioms that explain a certain entailment (cf Section 2.3).

For a more detailed introduction to lattice theory, the interested reader is referred to Grätzer [1998].

2.2.1 Lattices on Posets and Algebras

Lattices can be defined as special partially ordered sets (posets) or as algebraic structures. Both definitions can be shown to be equivalent.

Poset Definition

Let L be a finite set and let $\leq$ be an ordering relation on $L \times L$ which fulfills the following three properties for all $a, b, c \in L$:

(P1)	Reflexivity	$a \leq a$
(P2)	Antisymmetry	$a \leq b$ and $b \leq a$ imply that $a = b$
(P3)	Transitivity	$a \leq b$ and $b \leq c$ imply that $a \leq c$

The pair $\langle L; \leq \rangle$ is called a poset. If for a, b, the pair $\langle a, b \rangle$ or the pair $\langle b, a \rangle$ is part of the binary relation $\leq$, then a and b are said to be comparable w.r.t. $\leq$. If neither $\langle a, b \rangle \in \leq$ nor $\langle b, a \rangle \in \leq$, then a and b are said to be incomparable w.r.t. $\leq$, denoted by $a || b$.

An element $a \in L$ is said to be an upper bound for a subset $H \subseteq L$, if $h \leq a$ for all $h \in H$. The element a is said to be the least upper bound or the supremum of H, if for any $b \in L$ that is an upper bound for H we have that $a \leq b$. We define the greatest lower bound called the infimum of a subset $H \subseteq L$ analogously. If the supremum or the infimum exists, then they are unique. We refer to the infimum and the supremum of two elements a, b as $\sup\{a, b\}$ and $\inf\{a, b\}$, respectively.

Supremum and infimum of posets

For a lattice $\langle L; \leq \rangle$ there can exist two special kinds of supremum and infimum. For the empty set $\emptyset$ we refer to $\sup \emptyset$ as the zero element of L denoted by 0. Analogously, we refer to $\inf \emptyset$ as the unit element of L denoted by 1. In other words: if they exist, the smallest and the largest element of L w.r.t. $\leq$ are referred to as 0 and 1, respectively.

A poset $\langle L; \leq \rangle$ is called a lattice, if $\sup\{a, b\}$ and $\inf\{a, b\}$ exist for all $a, b \in L$. Furthermore, a poset $\langle L, \leq \rangle$ is a lattice, if for all non-empty subsets $H \subseteq L$ we have that $\sup H$ and $\inf H$ exist.

We use lattices to define properties on sets of elements—in our case sets of axioms. We define such a property as a set of tuples of the form $(I, \mathcal{T})$, where I is an input and $\mathcal{T}$ is a finite set of axioms, e.g. a TBox. We call these tuples an *axiomatized input*.

Algebraic Definition

To give an algebraic definition of lattices, we can define operators on pairs of axioms:

$$\begin{aligned} a \oplus b &= \inf\{a, b\} \\ a \otimes b &= \sup\{a, b\} \end{aligned}$$

We refer to $a \oplus b$ as the meet and to $a \otimes b$ as the join of a and b. Both are binary operations on a lattice $(L, \leq)$, i.e. they are functions on $L \times L \to L$. Both operations can be shown to fulfill the following properties:

(L1) Idempotency
$$\begin{aligned} a \otimes a &= a \\ a \oplus a &= a, \end{aligned}$$

(L2) Commutativity
$$\begin{aligned} a \otimes b &= b \otimes a \\ a \oplus b &= b \oplus a \end{aligned}$$

(L3) Associativity
$$\begin{aligned} (a \otimes b) \otimes c &= a \otimes (b \otimes c) \\ (a \oplus b) \oplus c &= a \oplus (b \oplus c) \end{aligned}$$

(L4) Absorption Identities
$$\begin{aligned} a \otimes (a \oplus b) &= a \\ a \oplus (a \otimes b) &= a \end{aligned}$$

An algebra $\langle L; \otimes, \oplus \rangle$ is called a lattice, if L is finite and non-empty, and $\otimes$ and $\oplus$ fulfill the properties $(L1) - (L4)$.

Diagram Representation

Posets (and hence lattices) can be represented by so-called *Hasse diagrams* as shown in Figure 2.1. For a lattice $\langle L; \leq \rangle$ the elements of L are represented as nodes. Two elements a, b are connected by a straight line, if one covers the other. An element $a \in L$ is said to cover an element $b \in L$, if $a < b$ and there does not exist any element $x \in L$ such that $a < x < b$. The binary order relation $<$ is defined as $x \leq y$ and $x \neq y$. If $a < b$, then the node for b is presented on a higher position than the node for a.

In the scope of this work we will only use the diagram representation for illustration purposes. There exist further methods for representing lattices which can be found in Grätzer [1998].

2.2.2 Lattices for Boolean formulas

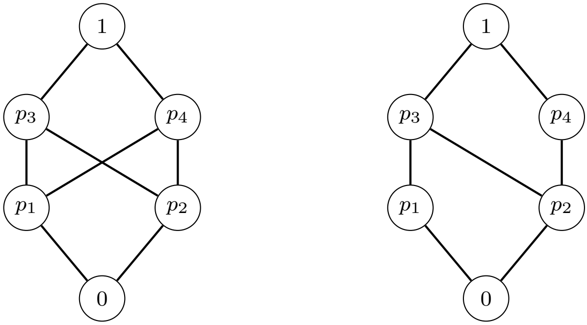
It turns out that the set of monotone Boolean formulas $\mathcal{B}$ over a finite set of propositional variables P forms a so-called *distributive lattice* $\mathbb{B}$ [Grätzer, 1998].[8] We uniquely assign each element of the lattice a monotone Boolean formula over P. Monotonicity ensures the partial order, i.e. if $\psi \rightarrow \phi$, then $\phi \leq \psi$ for two monotone Boolean formulas ϕ and ψ.

$$\phi_i \in \mathcal{B} \leftrightarrow p_i \in \mathbb{B}$$

The assignment is done using a labeling function $lab : \mathcal{B} \rightarrow \mathbb{B}$. For the sake of simplicity, we identify ϕ_i with $lab(p_i)$. For $\mathbb{B}$ to be a distributive lattice means that for any formulas $\phi, \psi, \rho \in \mathcal{B}$ it holds that $\rho \otimes (\phi \oplus \psi) = (\rho \otimes \phi) \oplus (\rho \otimes \psi)$. Figure 2.2 provides an example for a lattice over the set of two propositional variables $\{p_1, p_2\}$. The meet operator refers to conjunction whereas the join operator refers to disjunction. For example, $(p_1 \wedge p_2) \otimes (p_2 \wedge p_3) = (p_2)$ whereas $(p_1 \wedge p_2) \oplus (p_2 \wedge p_3) = (p_2 \wedge (p_1 \vee p_3)$. Special cases

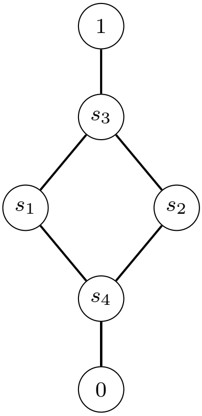
Distributive lattices

[8] Monotone Boolean formulas can be formed by conjunction and disjunction but not negation.



(a) **(b)**

Figure 2.1: The graph in the left Figure (a) is not a lattice, because p_1 and p_2 have no join. The graph on the right side in Figure (b) is indeed a lattice.



$$\begin{array}{lcc} lab(s_1) & = & \phi_1 \\ lab(s_2) & = & \phi_2 \\ lab(s_3) & = & \phi_1 \vee \phi_2 \\ lab(s_4) & = & \phi_1 \wedge \phi_2 \end{array}$$

Figure 2.2: Example for lattice over the set of all Boolean formulas over the set of two propositional variables $P = \{p_1, p_2\}$

are the global infimum 0 and the global supremum 1. For the set $\{p_1, p_2, p_3\}$ these result, for example, from $(p_1) \otimes (p_2) = 0$ and $(p_1 \oplus p_2 \oplus p_3) = 1$. Computation on the whole lattice $\mathbb{B}$ is not desirable, because the set of all Boolean formulas over P contains $2^{|P|}$ elements. To restrict ourselves to only a certain portion of a lattice, we can define sub-lattices. For a lattice $(L, \leq)$, the sub-lattice induced by the set $L' \subseteq L$ is defined as the closure of L' regarding join and meet, i.e. the induced sub-lattice is $\big((L'_{\oplus} \cup L'_{\otimes}), \leq\big)$.

2.3 Justifications

Every philosophical problem, when it is subjected to the necessary analysis and justification, is found either to be not really philosophical at all, or else to be, in the sense in which we are using the word, logical.

Bertrand Russell (1882-1970),
British philosopher,
mathematician and logician.

To *invalidate* an unwanted entailment $\mathcal{T} \models \eta$, i.e. removing η from $(\mathcal{T})^+$, we must first find out, which axioms are responsible for this very entailment. We are hence looking for the minimal sets of axioms $J \subseteq \mathcal{T}$ for which $J \models \eta$ holds [Schlobach and Cornet, 2003]. Minimality guarantees that the entailment is invalidated for this justification, when at least one of the axioms from the corresponding minimal set is not considered alongside with the others in $\mathcal{T}$ when drawing inferences. Furthermore, the entailment is invalidated w.r.t. $\mathcal{T}$, when it is invalidated w.r.t. to *all* its justifications.

Justifications from different perspectives

Justifications can be looked upon in different ways. One popular method is considering them as minimal sets of axioms. Besides the axiomatic view, we can also define them as monotone property on sets of monotone Boolean functions over a finite set of axioms. We will cover both views in Section 2.3.1 and 2.3.2. However, we use the axiomatic view in the remainder of this work, because our method is not based on lattices. For solving justifications we also have to deal with possible interdependencies between them. This is described in Section 2.3.3.

2.3.1 Justifications as Minimal Sets of Axioms

We call minimal sets of axioms from a TBox such that an inference still holds *justifications*:

Definition 5 (Justification)**.** *Let $\mathcal{T}$ be a TBox. A justification for an entailment $\mathcal{T} \models \eta$ is a set of axioms $J_{\mathcal{T},\eta} \subseteq \mathcal{T}$ such that $J_{\mathcal{T},\eta} \models \eta$ and $J' \not\models \eta$ for every $J' \subset J_{\mathcal{T},\eta}$.*

In order to be able to distinguish individual justifications, we number them by $k = 0, \ldots, K$ where K is the number of justifications. Consequently, we refer to the k-th justification by $J^k_{\mathcal{T},\eta}$.

Root justifications

Justifications for unsatisfiable concepts, called *JUC*, are of the form $J_{\mathcal{T},U\sqsubseteq\bot}$ where U is a (possibly complex) unsatisfiable concept. JUC may depend completely on the satisfiability of other concepts, i.e. they are supersets of other justifications. If a JUC does not depend on some other JUC, then we call it a *root JUC* [Scharrenbach et al., 2010c; Meyer et al., 2010].[9]

Definition 6 (Root and Dependent Justifications)**.** *Let $\mathcal{T}$ be a TBox. A justification for $J_{\mathcal{T},U\sqsubseteq\bot}$ is called a* derived JUC*, if there exists some concept U' for which $J_{\mathcal{T},U\sqsubseteq\bot} \supset J_{\mathcal{T},U'\sqsubseteq\bot}$. Otherwise $J_{\mathcal{T},U\sqsubseteq\bot}$ is called a* root JUC*.*

By invalidating a JUC $J_{\mathcal{T},U\sqsubseteq\bot}$, i.e. by performing some operation such that $J_{\mathcal{T},U\sqsubseteq\bot} \not\models U \sqsubseteq \bot$, it can happen that not all dependent JUC are invalidated as well. We will hence only consider

[9] In contrast to Kalyanpur et al. [2005] we do not define dependency and being root on concepts but on sets of axioms.

such operators for which invalidating a JUC $J_{\mathcal{T},U\sqsubseteq\perp}$ implies that all JUC that depend on $J_{\mathcal{T},U\sqsubseteq\perp}$ are invalidated, too.

Hence, invalidating all root JUC for unsatisfiable concepts will make every so far unsatisfiable concept satisfiable again. Referring to the properties we proposed in Section 1, a framework for conflict-free ontology evolution has to provide a method for automatically invalidating all root JUC for unsatisfiable concepts.

Example 2 (Root and derived justifications)**.**
Assume the following TBox:
$\mathcal{T} = \{B \sqsubseteq A, C \sqsubseteq B, C \sqsubseteq \neg A, D \sqsubseteq C, D \sqsubseteq \neg A, E \sqsubseteq D\}$

The concepts C, D and E are entailed unsatisfiable. The only justification for $\mathcal{T} \models C \sqsubseteq \perp$ is
$J^0_{\mathcal{T},C\sqsubseteq\perp} = \{B \sqsubseteq A, C \sqsubseteq B, C \sqsubseteq \neg A\}$

The concept D is also inferred unsatisfiable, but with the following two justifications:

$$\begin{aligned} J^1_{\mathcal{T},D\sqsubseteq\perp} &= \{B \sqsubseteq A, C \sqsubseteq B, D \sqsubseteq C, D \sqsubseteq \neg A\} \\ J^2_{\mathcal{T},D\sqsubseteq\perp} &= J^0_{\mathcal{T},C\sqsubseteq\perp} \cup \{D \sqsubseteq C\} \end{aligned}$$

As such, $J^1_{\mathcal{T},D\sqsubseteq\perp}$ is a root JUC whereas $J^2_{\mathcal{T},D\sqsubseteq\perp}$ is derived.
For the unsatisfiability of the concept E there exist the two justifications: $J_{\mathcal{T},E\sqsubseteq\perp} = J^1_{\mathcal{T},D\sqsubseteq\perp} \cup \{E \sqsubseteq D\}$ and $J_{\mathcal{T},E\sqsubseteq\perp} = J^2_{\mathcal{T},D\sqsubseteq\perp} \cup \{E \sqsubseteq D\}$. Both justifications for $\mathcal{T} \models E \sqsubseteq \perp$ are derived.

2.3.2 The Pinpointing-Formula

Another view on justifications is to consider them as a minimal set of axioms such that a certain property $\mathcal{P}$ is still fulfilled. For a TBox $\mathcal{T}$ and a set $\mathcal{I}$ we speak of a *monotone axiomatized input* $(\mathcal{T},\mathcal{I}) \in \mathcal{P}$, if $(\mathcal{T},\mathcal{I}) \in \mathcal{P} \Leftarrow (\mathcal{T}',\mathcal{I}) \in \mathcal{P}$ for all $\mathcal{T}' \supset \mathcal{T}$ [Baader and Peñaloza, 2008]. There exist various such properties and inputs, such as unsatisfiability of concepts and monotone Boolean formulas over a finite set of propositional variables, respectively.

Justifications can be represented as monotone Boolean formulas over a finite set of propositional variables. In this case these

Justifications as monotone Boolean formulas

monotone Boolean formulas act as an indicator function: We assign each axiom τ_i of a TBox $\mathcal{T} = \{\tau_0, \ldots, \tau_I\}$ a unique propositional variable p_i. The finite set of propositional variables is denoted by P. *Valuations* $\mathcal{V}$ of the set of monotone Boolean formulas over P induce subsets $\mathcal{T}_\mathcal{V}$ of the TBox $\mathcal{T}$:[10]

$$\mathcal{T}_\mathcal{V} = \{\tau_i \in \mathcal{T} \quad | \quad p_i \text{ is } true \text{ in } \mathcal{V}\}$$

We consider monotone Boolean functions over P. Hence, if for two of these formulas ψ, ϕ it holds that $\psi \rightarrow \phi$, then any valuation that satisfies ψ must also satisfy ϕ. This corresponds to the monotonicity of the standard inference operator on TBoxes: If $\mathcal{T}_1 \supset \mathcal{T}_2$, then for all η for which $\mathcal{T}_1 \models \eta$ it holds that also $\mathcal{T}_2 \models \eta$.

The pinpointing formula

In the case of justifications for an entailment η, we are interested in *minimal* subsets of a TBox $\mathcal{T}$ such that $\mathcal{T} \models \eta$. We hence have to find a monotone Boolean formula ϕ over P such that for any valuation $\mathcal{V}$ that satisfies ϕ,[11] the entailment is still valid w.r.t. the induced TBox $\mathcal{T}_\mathcal{V}$.[12] This formula is called the *pinpointing formula* for η.

It turns out that the pinpointing formula encodes all justifications for η. A justification $J_\eta = \{\tau_{i_0}, \ldots, \tau_{i_J}\}$ for an entailment η is nothing but a special subset of $\mathcal{T}$. It corresponds to a conjunction of the variables $p_{i_0}, \ldots, p_{i_J}$ that belong to the axioms $\tau_{i_0}, \ldots, \tau_{i_J}$, where $J_\eta^k = \{\tau_{i_0}, \ldots, \tau_{i_J}\}$:

$$J_\eta^k = \{\tau_{i_0}, \ldots, \tau_{i_J}\} \quad \sim \quad \bigwedge_{j=0}^{J} p_{i_j}$$

[10] A valuation is a function that assigns each element of a finite set of propositional variables P a truth value.

[11] A valuation $\mathcal{V}$ satisfies a Boolean formula ϕ, if the result of ϕ is $true$ when assigning each variable in ϕ the truth value determined by $\mathcal{V}$

[12] $\mathcal{T}_\mathcal{V} = \{\tau \in \mathcal{T} | lab(\tau) \in \mathcal{V}\}$

The set of all justifications for an entailment η can then be represented as the disjunction of these conjunctions:

$$\{J_\eta^k\}_{k=0}^K \quad \sim \quad \phi_\eta = \bigvee_{k=0}^{K} \left(\bigwedge_{j_k=0}^{J_k} p_{i_{j_k}} \right)$$

For the monotone Boolean formula ϕ_η that encodes all justifications for η w.r.t. $\mathcal{T}$, each TBox $\mathcal{T}_\mathcal{V} \subseteq \mathcal{T}$ that is induced by a valuation $\mathcal{V}$ of ϕ_η entails η as a result:

$$\mathcal{V} \models \phi_\eta \quad \Longleftrightarrow \quad T_\mathcal{V} \models \eta$$

The *pinpointing formula* can then be defined according to Baader and Peñaloza [2008]:

Definition 7 (Pinpointing Formula). *Let $\mathcal{T}$ be a TBox and η an entailment $\mathcal{T} \models \eta$. For each axiom $t_i \in \mathcal{T}$ let p_i be a unique propositional variable. A pinpointing formula ϕ for $\mathcal{T} \models \eta$ is a monotone Boolean formula over $P = \{p_0, \ldots, p_I\}$ such that for each valuation $\mathcal{V}$ of P, $\mathcal{V} \models \phi$ iff $\mathcal{T}_\mathcal{V} \models \eta$.*

If we set $\eta = (U_0 \sqcup \ldots U_N) \sqsubseteq \bot$ for all concepts U_n that are unsatisfiable w.r.t. $\mathcal{T}$, then the pinpointing formula for η corresponds to the set of all root JUC for $\mathcal{T} \models U_n \sqsubseteq \bot$ for $n = 0, \ldots, N$. This can be seen as follows. For the entailment $\eta_0 = U_0 \sqsubseteq \bot$, the pinpointing formula for η_0 corresponds to all JUC (also the dependent w.r.t. $\mathcal{T}$) for $\mathcal{T} \models U_0 \sqsubseteq \bot$.

The pinpointing formula and root JUCs

For $\eta_1 : (U_0 \sqcup U_1) \sqsubseteq \bot$, the pinpointing formula corresponds to all JUC for U_0 as well as all JUC for U_1. If there exists $J^0_{\mathcal{T},U_1 \sqsubseteq \bot}$ that depends on $J^1_{\mathcal{T},U_1 \sqsubseteq \bot}$, then the first JUC is not minimal w.r.t. $\mathcal{T} \models (U_0 \sqcup U_1) \sqsubseteq \bot$. As a consequence, the dependent JUC for U_0 and U_1 have no counterpart in the pinpointing formula. We can hence show by induction that the pinpointing formula for $\eta : (U_0, \ldots, U_N) \sqsubseteq \bot$ corresponds to root JUC only.

2.3.3 Representing Dependencies Between Justifications

As we saw in Section 2.3.1, justifications may depend on each other. We also saw that solving the independent, i.e. the root JUC can solve all conflicts. Hence, we have to define a structure on JUC that provides the dependency information. As we could see in Section 7, we can represent the set of root JUC by a pinpointing formula. Indeed the set of Boolean formulas that correspond to JUC is a sub-lattice of the (distributive) lattice $\mathbb{B}$. We hence define the justification lattice as follows:

Let $\mathcal{T}$ be a TBox and $p_0, \ldots, p_M$ be a set of unique propositional variables on the axioms $\tau_0, \ldots, \tau_M$ such that $lab(p_i) = \tau_i$. We define for each JUC $J^k_{U \sqsubseteq \bot}$ for $\mathcal{T}$ the monotone Boolean formula $\phi_k = \bigwedge p_i$ such that $p_i \in \phi_k$ iff $\tau_i \in J^k_{U \sqsubseteq \bot}$ and $J^k_{U \sqsubseteq \bot} = \bigcup_{p_i \in \phi_k} lab(p_i)$.

Let $\mathcal{J} = \bigcup_{k=0}^{K} J^k_{U \sqsubseteq \bot}$ be the set of all JUC for $\mathcal{T}$ and let $\mathcal{J}_\mathcal{B}$ be the set of all corresponding monotone Boolean formulas ϕ_k. The pair $\langle \mathcal{J}_\mathcal{B}; \otimes, \oplus \rangle$ is the lattice for all JUC w.r.t. $\mathcal{T}$. It is equal to the lattice $\langle \mathcal{J}; \leq \rangle$ up to isomorphism, where the partial order $\leq$ refers to subset inclusion. The root JUC then refer to the least elements in $\langle \mathcal{J}; \leq \rangle$. Please note that the poset $\langle \mathcal{J}; \leq \rangle$ neither has to have a 0-element nor a 1-element. It can easily checked that $\langle \mathcal{J}; \leq \rangle$ is indeed a lattice.[13]

In the following, we will refer simply to $\mathcal{J}$ as the lattice on all JUC w.r.t. $\mathcal{T}$

[13] If this was not the case, there had to exist an JUC that is the union of two justifications. It can be shown that for such a union of two justifications, one of these justifications has to include to other.

2.3.4 Laconic Justifications

It is possible that only parts of axioms are responsible for certain entailments. Laconic justifications [Horridge et al., 2008] provide a way to define justifications on parts of axioms only. Therefore, the axioms become subject to a structural transformation that transforms them into several, weaker but simpler axioms.

Example 3 (Laconic justifications)**.** *Let* $\mathcal{T} = \{B \sqsubseteq A_1 \sqcap \neg A_2, C \sqsubseteq B, C \sqsubseteq \neg A_1\}$*. Obviously,* $\mathcal{T} \models C \sqsubseteq \bot$ *with the only JUC* $J^1_{C \sqsubseteq \bot} = \mathcal{T}$*. However, we can transform the axiom* $B \sqsubseteq A_1 \sqcap A_2$ *into the pair of axioms* $B \sqsubseteq A_1$ *and* $B \sqsubseteq \neg A_2$*. In that case we receive a laconic justification* $J^2_{C \sqsubseteq \bot} = \{B \sqsubseteq A_1, C \sqsubseteq B, C \sqsubseteq \neg A_1\}$*. The information that* $B \sqsubseteq \neg A_2$ *is not needed in this case. We can hence perform a more fine-grained repair on* $\mathcal{T}$ *for solving the unsatisfiability of* C*.*

Laconic justifications hence consist, in general, not only of axioms from an ontology $\mathcal{O}$, but also of axioms of its deductive closure $(\mathcal{O})^*$. In the example above, the axiom $B \sqsubseteq A_1$ is part of a laconic justification for $\mathcal{T} \models C \sqsubseteq \bot$. Yet this axiom is not contained in $\mathcal{T}$, but in $(\mathcal{T})^*$. To ensure that only meaningful axioms can be drawn from the deductive closure for laconic justifications,[14] the choice can be restricted to a subset of the deductive closure.

Laconic justifications

There exist many more rules for making regular justifications laconic as well as exist more approaches towards fine-grained justifications. The interested reader may find a good overview on the subject in Horridge et al. [2008].

[14] In the example above, the set $\{C \sqsubseteq B \sqcap \neg B\}$ is also a laconic justification. However, we could not use it for resolving $\mathcal{T} \models C \sqsubseteq \bot$.

2.4 Contradictions

The principle that two contradictory statements are not both true is the most certain of all.

Aristotle (384 BC-322 BC)
Greek philosopher

Contradictions in classic logics

Quoted after Łukasiewicz [1971] Aristotle's statement formulates *the logical principle of contradiction*. While being the foundation of many modern formal logic systems, already several ancient logicians, including Heraclitus of Ephesus, Antisthenes the Cynic, and others opposed to Aristotle's view. Accordingly, more current philosophers and logicians picked up on this opposing view which resulted in the development of several new logic systems in the 19th and 20th century. For example, in Hegel's *dialectics* co-existence of two contradictory statements is possible [Hegel, 1991]. Accordingly, with the foundation of modern mathematical logics in the 20th century, logical systems were investigated on an increasing level of precision. More and more contradictions were found in theories that we widely accepted to be true. One famous example for that is Russel's antinomy [Russell, 1937] which proved that the naive set-theory—one of the foundations of mathematics in that time—leads to a contradiction.

However, transferring Aristotle's principle of contradiction to modern formal logics risks a misconception. Aristotle's logic is based on judgments and concepts—whereas in mathematical logic reference is made to formulas (also called sentences) and terms. Instead of expressing Aristotle's principle of contradiction as: "Two contradictory sentences are not both true.", we have namely to add: "in the same language" or "if the words occurring in those sentences have the same meanings" [Jaśkowski, 1999]. In the case of DL, these words are concepts, roles, and assertions and sentences are axioms and assertions.

We refer to drawing logical consequences based on Aristotle's principle of contradiction as *standard reasoning* and to those that do not as *non-standard reasoning*. In the following we define what contradictions w.r.t. standard-reasoning are and what happens to them when we apply non-standard reasoning. For further information about non-standard reasoning relevant to this work, the reader is referred to the related work in Section 8.2.5.

Non-standard reasoning

2.4.1 Modeling Errors in Description Logics

Contradictions can occur in any part of a DL knowledge base. While we present an overview over possible contradictions for all parts of a DL knowledge base, this work focuses on solving contradictions in the TBox solely.

Contradictions in the TBox

We learned in Section 2.1 that Description Logics are a family of decidable fragments of first-order logics. First-order logics fulfill Aristotle's principle of contradiction, and hence a contradiction in a DL TBox occurs, when, for example, a concept U is entailed to be subsumed by a concept and its complement. That is the case when $U \sqsubseteq A$ and $U \sqsubseteq \neg A$ are both logical consequences from a TBox $\mathcal{T}$. In this case we have that $U^{\mathcal{I}} = \emptyset$, i.e. U is unsatisfiable. Hence a concept that is unsatisfiable w.r.t a TBox refers to a contradiction.

Definition 8 (TBox Contradictions)**.** *A TBox $\mathcal{T}$ is said to contain a contradiction, if for any concept U in the signature of $\mathcal{T}$ it holds that $\mathcal{T} \models U \sqsubseteq \bot$. A TBox that does not contain a contradiction is called* coherent.

Unsatisfiable concepts may cause the concept hierarchy to become inconsistent, i.e. performing the classification task might lead to undefined results. If for a TBox $\mathcal{T}$ and a concept U we have that $\mathcal{T} \models U \sqsubseteq \bot$, then U is equivalent to the bottom concept $\bot$. In particular, $U \sqsubseteq \bot$, which together with $\mathcal{T} \models \bot \sqsubseteq C$ for any concept C by definition, implies $\mathcal{T} \models U \sqsubseteq C$. Hence, U is entailed to be subsumed by any concept C.

Example 4 (Concept hierarchy for unsatisfiable concepts)**.**
Let $\mathcal{T} = \{B \sqsubseteq A, C \sqsubseteq B, C \sqsubseteq D, D \sqsubseteq \neg A, G \sqsubseteq F\}$*. Obviously* $\mathcal{T} \models D \sqsubseteq \bot$*. Intuitively, we would like to have* $\mathcal{T} \models D \sqsubseteq B$*, but not* $\mathcal{T} \models D \sqsubseteq F$*. Yet, both entailments are valid.*

Inconsistent concept hierarchies can cause problems when we try to determine the deductive closure of a TBox in Section 7.2.

Contradictions in the RBox

Contradictions in an RBox refer to violations of the role hierarchy or violations of role assertions. This happens, for example, if we assert a role R to be reflexive and state that another role S is irreflexive but subsumed by R.

Definition 9 (RBox Contradictions)**.** *An RBox* $\mathcal{R}$ *is said to contain a contradiction, if for any role* R *in the signature of* $\mathcal{R}$ *it holds that* $\mathcal{R} \models R \sqsubseteq \bot$*. An RBox that does not contain a contradiction is called* coherent.

Although we do not explicitly cover solving RBox contradictions in the scope of this thesis, we give a formal definition, since we believe that the approach used in this work can also be applied to the case of RBox contradictions.

Contradictions in the ABox

There are several types of contradictions for ABoxes, also referred to as *clash* or *inconsistency*. Either type can cause the entailment relation not to deliver any useful information anymore.

The first type of inconsistency happens, when for an assertion $A(b)$ that is entailed by a knowledge base, also its complement

$\neg A(b)$ is entailed. If we assert, for example, in an ABox $\mathcal{A}$ the instance b to the concept U, and if U is unsatisfiable w.r.t. the TBox $\mathcal{T}$, then $\mathcal{T}$ and $\mathcal{A}$ have $C(b)$ as a logical consequence for any concept C—also for complex concepts like $\neg C$. Hence we have that $\mathcal{T}$ and $\mathcal{A}$ have $C(b)$ and $\neg C(b)$ as a logical consequence for any concept C which, in turn, refers to a contradiction.

Types of inconsistencies

Further types of contradictions in ABoxes are related to roles. These can be violations of cardinality restrictions and datatype range restrictions on roles, but also violations of role assertions. In the first case, the domain or range of a property is restricted to a maximal cardinality but is asserted more than the admissible number of distinct individuals. In the second case, we have, for example, restricted the range of a role R to positive integers but assert $R(a, -1)$ for an individual a. In the third case we violate, for example, a functional role assertion (i.e. that an individual must not be asserted more than one distinct individual for that role) for a role R, if the following two assertions $R(a, b)$ and $R(a, c)$ are contained in the ABox for the distinct individuals a, b, and c. Since we do not investigate ABox contradictions in depth in this thesis, we omit a formal definition.

2.4.2 Exceptional Information

Human agents do not tend to think in strict consistent DL models based on classical logics: in our mind we allow certain things to be ambiguous. This can be, for example, exceptions to a general case. As such, exceptions refers to contradictions in DL-knowledge bases for standard-reasoning, because we can only express that something is either true or not. The penguin example below provides an excellent illustration for exceptional knowledge.

Example 5 (Flying penguins). *Assume we know that birds are flying animals. Assume further that we also know that penguins are birds. However, penguins cannot fly, hence they are not flying birds. If we express this knowledge in DL using the concepts $FlyingAnimal$, $Bird$ and $Penguin$, we can express out knowledge about the world as:*

$$\begin{aligned}\mathcal{T}_{penguins} &= \{Bird \sqsubseteq FlyingAnimal, Penguin \sqsubseteq Bird,\\ &\quad Penguin \sqsubseteq \neg FlyingAnimal\}\end{aligned}$$

It is not hard to see that $\mathcal{T}_{\text{penguins}} \models Penguin \sqsubseteq \bot$. The concept $Penguin$ is unsatisfiable, because $\mathcal{T}_{\text{penguins}}$ entails, on the one hand, that an instance of $Penguin$ is also an instance of $FlyingAnimal$. On the other hand, we explicitly stated that this is not the case.[15] Hence we have a conflict according to Aristotle's logical principle of contradiction.

Making Exception Explicit

We could solve this conflict by introducing two additional concepts for birds, i.e. $FlyingBird$ and $NonFlyingBird$ and declare them as a subclasses of the original concept $Bird$. Accordingly, we would have to change the axiom $Penguin \sqsubseteq Bird$ to the axiom $Penguin \sqsubseteq NonFlyingBird$ and add the axiom $NonFlyingBird \sqsubseteq FlyingAnimal$. Yet, this approach has several disadvantages:

First, we have add additional information, as well as to remove existing information. Furthermore, finding a meaningful name for the new concepts can only be performed manually. Last, in case there are more axioms involved, the choice of the concept to be replaced is ambiguous. Hence we violate some of the properties that a repair operator should meet: We modify original information and cannot work fully automatically.

[15] Note that $Penguin \sqsubseteq \neg FlyingAnimal$ does not have to be stated explicitly but may also be entailed implicitly. In any case $Penguin$ is entailed unsatisfiable by $\mathcal{T}_{\text{penguins}}$.

Besides violating these properties the solution, though being consistent, is still not capable of expressing preference. In case we assert some individual b to be a $Bird$, then we can at most b is a $FlyingBird$ or $NonFlyingBird$ as an entailment—we are not able to prefer the assertion $NonFlyingBird(b)$ in classical logics.

Changing the Semantics

If we do not want to change the knowledge representation for exceptional information, we have to change the semantics. Instead of modeling every case explicitly, we would like to generalize certain pieces of information while allowing for exceptions from general rules. The modeling error from Example 5 is not considered as erroneous information but as a valid view on the world. As we saw above, strict FOL entailment cannot be applied anymore. In the case of exceptional information we can apply, for example, Default Logics (Section 2.5), which were designed to enable entailment on exceptional information.

How to keep the knowledge representation.

Exceptional information, in turn, always refers to contradictions in strict DL.[16] If we want to resolve an exception in DL, we have to explicitly state all the exceptional cases. From a Default Logics perspective, an exception can be used to allow for a more compact representation without having to specify all the exceptional cases. In Example 5, we can express the fact that there exists an exception, i.e. the penguins, to the axiom that all birds are flying animals by treating certain axioms separately for entailment.

[16] But not every set of Defaults contains a contradiction. This is the case when there exists only one partition $\mathcal{U}_0$ with an empty remainder set D_0 (cf Section 2.5).

Splitting sets of axioms is the key.

Splitting the set of axioms means invalidating entailments w.r.t. the whole knowledge base.[17] We do not have to split up the concept of $Bird$ into $FlyingBird$ and $NonFlyingBird$, but consider entailment on the following two TBoxes separately:

$$\begin{aligned}\mathcal{T}_1 &= \{Bird \sqsubseteq FlyingAnimal\}\\ \mathcal{T}_2 &= \{Penguin \sqsubseteq Bird, Penguin \sqsubseteq \neg FlyingAnimal\}\end{aligned}$$

We learn that the axiom $Penguin \sqsubseteq \bot$ is no longer entailed, because we invalidated $Penguin \sqsubseteq FlyingAnimal$. In contrast to axiom removal, we are able to keep all axioms but we have to accept that certain—possibly important—entail-ments are not valid anymore.

2.4.3 Perpendicular Views on the World

Different People can have different contradicting conceptualizations of the world. Especially when they are from different domains or have a different cultural background, but also experts within a domain may model the concept hierarchy of a domain in such a way that such taxonomies contradict each other. In quantum mechanics, for example, the electron is conceptualized in two different ways: On the one hand an electron is a particle, on the other hand it is considered a wave. Yet, waves are not particles and vice versa. The explicit nature of the axioms make it impossible to consider both of them at the same time—they are mutually exclusive, but nevertheless hold in a certain view on the world. We will see in Section 5.2 how such contradictions may be solved.

[17] We will give a formal definition on invalidation of entailments in Section 3.1.

2.5 Default Logics

The need to make default assumptions is frequently encountered in reasoning'about incompletely specified worlds. Inferences sanctioned by default are best viewed as beliefs which may well be modified or rejected by subsequent observations. It is this property which leads to the non-monotonicity of any logic of defaults.

Raymond Reiter (1939-2002)
Canadian computer scientist and logician

In contrast to classical monotonic Aristotelian logics, *Default Logics* are able to express that *something is true by default*. The situation in Example 5 represents a situation where birds are animals that can *typically* fly. Classical logics are not able to express exceptions to a rule besides specifying all exceptions to this rule, explicitly. In the example, we would have to introduce the concepts of flying and non-flying animals.

Default Logics as introduced originally by Reiter [1980] express rules in the form:

$$\frac{P : J_0, \ldots, J_N}{C}$$

Assume we have a set of current facts W that we belief to be true. If we *believe* the so-called prerequisite P to be true and if, furthermore, the justifications J_0, ..., J_N are consistent with our current beliefs, then we also belief that the conclusion C is true. In our example, we could express the fact that typically, all birds with wings fly by the default rule:

$$\frac{Bird(x) : HasWings(x)}{Fly(x)}$$

A Default theory is a pair $\langle D, W \rangle$ formed by the current beliefs W and the default rules, called defaults, D.

2.5.1 Lehmann's Default Logics

Another perspective on Defaults

However, the deductive closure of Default theories is hard to compute and the consequence relation is not rational, i.e. does not meet the requirements for the rational monotonicity property [Lehmann and Magidor, 1992]. Lehmann therefore introduced another perspective on the semantics of defaults which restricts the definition to *normal defaults* [Lehmann, 1995]. Normal defaults require that justification and conclusion are the same:

$$\frac{P : C}{C}$$

The inference operator is required to be rational, i.e. to fulfill the rational monotonicity property. He further based the inference operator on lex-minimal models which will be explained below. That way, we can stay with (the more simple) normal defaults while being able to express knowledge like "With the exception of penguins, birds fly, typically".

Lehmann proposed the following four properties that an inference operator for useful Default Logics should meet:

1. the presumption of typicality,
2. the presumption of independence,
3. priority to typicality, and
4. respect for specificity.

In Lehmann's Default Logics the axioms are not changed but interpreted in a different way. Let $\mathcal{K}$ be a finite set of first-order logics rules where each rule is of the form $b \rightarrow a$. We call b the *presumption* and a the *conclusion*. If we consider this axiom under Default semantics, we refer to it as the *default* $b : a$.

The semantics of a default $b : a$ differs from standard semantics in the way that we allow contradicting defaults. Conflicts are resolved by preferring more specific defaults over more general defaults [Baader and Hollunder, 1993]. In Lehmann's Default Logics, the set of defaults is divided into *partitions* $\mathcal{U}_0, \ldots, \mathcal{U}_N$. The defaults in partition $\mathcal{U}_{n+1}$ model *more specific information* than the defaults in the previous partition $\mathcal{U}_n$.

Partitions are computed as follows [Lehmann, 1995]: The first partition $\mathcal{U}_0$ is the set of all rules $b \rightarrow a$ in $\mathcal{K}$ such that there exists a model for $\mathcal{K}$ for which both, b and a are true. We remove the partition from the set of rules and obtain the remainder set $D_1 = \mathcal{K} \setminus \mathcal{U}_0$. We proceed in the same way and compute the next partition $\mathcal{U}_1$ for the remainder set D_1. Starting with $D_0 = \mathcal{K}$ we inductively construct partitions $\mathcal{U}_0, \ldots, \mathcal{U}_N$.

Example 6 (Partitions). *In Example 5 the only axiom for which the concepts on both sides are satisfiable is $Bird \sqsubseteq FlyingAnimal$. Consequently, we have that $\mathcal{U}_0 = \{Bird \sqsubseteq FlyingAnimal\}$ and $D_1 = \{Penguin \sqsubseteq Bird, Penguin \sqsubseteq \neg FlyingAnimal\}$.*

Regarding D_1, we have for all axioms, that either side of the axiom is a concept that is satisfiable w.r.t. D_1. Hence $\mathcal{U}_1 = D_1$ and $D_2 = \emptyset$.

Since the number of axioms was finite, the procedure terminates when it produces an empty partition. We hence come up with an ordered family of subsets of $\mathcal{K}$ that form a partition of $\mathcal{K}$:

$$\mathcal{K} = \mathcal{U}_0 \oplus \ldots \mathcal{U}_N \oplus D_N$$

The last remainder set D_N does not have to be empty, i.e. there exist rules in $\mathcal{K}$ for which we cannot find a model in which both, presumption and conclusion are true—even when we remove all other rules. This is, for example, the case for rules of the form $b \rightarrow a \wedge \neg a$. In that case, the Default knowledge base is said to be inconsistent.

Partitions can be computed in time $\mathcal{O}(n^2)$ where n is the number of axioms in $\mathcal{K}$.

2.5.2 Default Logics Entailment

Models for Default Logics

The extension, i.e. the deductive closure of a Default knowledge base is computed according to so-called *lex-minimal models*. For each model a vector is defined, which is sorted in a lexicographical order, and valid inferences correspond to lexicographical minimal models. This order corresponds to a preference relation over possible models such that models that violate the least number of the least impacting defaults, i.e. less specific defaults, are preferred over others. This reflects the desired properties defined for Leh-mann's Default Logics: the presumption of typicality, the presumption of independence, priority to typicality, and respect for specificity.

Let $\mathcal{I}$ be an interpretation. For a possible entailment η, we count the number defaults that $\mathcal{I}$ violates, i.e. all $(b : a)$ for which $b \wedge \neg a$ is true under $\mathcal{I}$. We denote the number of violated defaults for each partition $\mathcal{U}_n$ by u_n. This results in an N-dimensional vector of positive integers. We can now sort this vector from right to left in lexicographical order, i.e. we first sort by dimension N, then by dimension $N - 1$ etc.

Example 7 (Lex-minimal models)**.** *Let, for example,* $[0, 10, 0]$, $[100, 0, 0]$ *and* $[0, 0, 1]$ *be the vectors of three different models for a default knowledge base that has three partitions. The lex-minimal order on these models is*

$$[100, 0, 0] \quad <_{lex-min} \quad [0, 10, 0] \quad <_{lex-min} \quad [0, 0, 1]$$

Comparing models in lexicographical order.

In Example 7 model $[0, 0, 1]$ is the least preferred. It indeed has the least number of defaults violated in total, but it violates the largest number of constraints in $\mathcal{U}_2$, i.e. in the most specific partition. Comparing the other two models, they violate equally many defaults in partition $\mathcal{U}_2$, so we have to compare them according to the next most specific partition, i.e. $\mathcal{U}_1$. Consequently, the model $[100, 0, 0]$ is preferred over $[0, 10, 0]$, because it violates less defaults in partition $\mathcal{U}_1$. Because the number of defaults violated differs already in partition $\mathcal{U}_1$, the lex-minimal order does not consider partition $\mathcal{U}_0$ in comparing the two models.

The preference relation induced by lex-minimal models does not make preference statements about concepts but about axioms. Example 7 shows how violating more specific axioms is considered more severe than violating more general axioms. Considering the penguins Example 5, the axiom that birds are flying animals might fall into the first partition, whereas the axioms that penguins are birds and penguins are not flying animals are contained in the second partition. Hence a model in which there are 100 instances of birds that cannot fly (all of which violate the axiom in the first partition is preferred over a model in which 10 instances of penguin are not instances of bird.

2.5.3 Default Logics and Description Logics

We can consider DL axioms under Default semantics. As we saw in Section 2.1.6, Description Logics are a fragment of first-order logics. Any subsumption axiom $B \sqsubseteq A$ can be identified with the FOL formula $\forall x : B(x) \rightarrow A(x)$. Hence, we define $(A|B)$ to be the Default for the subsumption axiom $B \sqsubseteq A$. Logical consequences can then be defined according to Default Logics entailment as defined by Lehmann's Lexicographical Entailment which was described in the previous section. Roughly speaking, by checking whether a Default $(A|B)$ is violated under some interpretation $\mathcal{I}$, we effectively check whether the assert $(A \sqcup \neg B)(i)$ is a logical consequence for an individual $i \in \Delta^{\mathcal{I}}$. More information on entailment on DL TBoxes w.r.t. Lehmann's Default Logics can be found in Lukasiewicz [2008].

We will show how Default logics can be applied to Description Logics for solving conflicts in Chapter 4.

2.5.4 Probabilistic Description Logics

Extending Default Logics

So far, Default Logics was only capable of defining the partition on the whole set of axioms, i.e. all axioms were considered as Defaults. As a consequence, all Defaults $(A|B)$ for which we can find a model that does not violate $(A|B)$ w.r.t. all other Defaults are added to the first partition $\mathcal{U}_0$. However, there can exist Defaults $(C|D)$ that are not violated by any model. We can collect these in a separate TBox, that we refer to as the *Universal TBox* $\mathcal{T}_\triangle$. For Probabilistic Description Logics, Lukasiewicz [2008] extended Lehmann's Default Logics with such a Universal TBox.[18] This separation allows to distinguish between those parts of a knowledge base that should be considered as Defaults and those parts that should be considered according to classical DL.

Probabilistic Description Logics provides a way how to partition the TBox according to Lehmann's Default Logics that we described in the previous section. This partitioning is the basis for solving contradictions with the operators that we describe in Chapters 4 and 5.

Partitions of axioms

A partition $\mathcal{U}_n$ is inductively defined as the set of axioms $D \sqsubseteq C$ from the remainder set D_n for which we can find a model that satisfies both C and D.[19] The remainder set is, in turn, inductively defined as $D_n = \mathcal{T}_\triangle \cup (\mathcal{T} \setminus \bigcup_{m=0}^{n-1} \mathcal{U}_m)$ where $\mathcal{T}_\triangle$ is a TBox that holds all those axioms that have to be satisfied at any time.[20] If the last remainder set D_n is empty, then the Default TBox $\langle \mathcal{T}_\triangle, (\mathcal{U}_0, \ldots, \mathcal{U}_N)\rangle$ is coherent.[21]

We should note that for computing the partitions we check whether for an TBox axiom we can find a model such that both

[18] The term *Universal TBox* is introduced within the scope of this thesis to be able to distinguish this very TBox $\mathcal{T}_\triangle$ from other TBoxes. In Lukasiewicz [2008] $\mathcal{T}_\triangle$ is just referred to as a regular TBox.

[19] Lukasiewicz calls this a model that verifies the constraint $(C|D)$.

[20] We will refer later on to $\mathcal{T}_\triangle$ as the Universal TBox.

[21] In the context of Lukasiewicz the pair $\langle \mathcal{T}_\triangle, (\mathcal{U}_0, \ldots, \mathcal{U}_N)\rangle$ is referred to a probabilistic TBox which is called consistent (instead of coherent), if the partitioning $\mathcal{U}_0, \ldots, \mathcal{U}_N$ exists.

sides of the axiom are satisfied. We hence perform satisfiability checks w.r.t complex concepts rather than for atomic concept names.

We omit a more detailed description of Probabilistic Description Logics and refer the interested reader to Lukasiewicz [2008].

Part II

Methods for Solving Conflicts

3

Invalidating Entailments

To resolve unsatisfiable concepts, we have to find a way how to make entailments of the kind $\mathcal{T} \models U \sqsubseteq \bot$ invalid for those U that are in the signature of $\mathcal{T}$. As there exist various ways how to accomplish this, we introduce the general Δ-Operator for invalidating entailments. We show how the methods for invalidating entailments can be defined by this operator whereby the different approaches become comparable.

Plan of This Chapter

We introduce the general Δ-Operator for invalidating entailments in Section 3.1. After defining the simplemost Δ-Operator possible in Section 3.2 we show in Section 3.3 how Ontology Repair can be defined in terms of the Δ-Operator. Continuing with presenting the data structures for those Δ-Operators that result in so-called Multi-View TBoxes we introduce the so called Splitting-Δ-Operator for these data structures in Section 3.4. It serves as the basis for many of the Δ-Operators we investigate later on in Chapter 4. We close this chapter with an investigation how the Δ-Operator fits into the context of belief change in Section 3.5.

3.1 A General Operator for Invalidating Entailments

We want to introduce an operator such that when we apply this operator on a TBox, the resulting structure does not contain entailments of the form $U \sqsubseteq \bot$ for any U in the signature of $\mathcal{T}$. We therefore require an entailment relation for the range of this operator. Before we define the operator in more detail, we formalize the process of invalidating entailments:

Definition 10 (Invalidating Entailments)**.** *Let $\mathcal{T}$ be a TBox and $\eta \in (\mathcal{T})^\star$. Applying an operator Δ to $\mathcal{T}$, we say the Δ invalidates an entailment $\eta \in (\mathcal{T})^\star$, if $\eta \notin (\Delta(\mathcal{T}))^\star$.*

Note that we do not require $(\Delta(\mathcal{T}))^\star$ to be a subset of $(\mathcal{T})^\star$. In fact, for example when we perform axiom rewriting, $(\Delta(\mathcal{T}))^\star$ may contain new axioms that are not contained in $(\mathcal{T})^\star$. The purpose of Definition 10 is to give a formal definition for the set of entailments we lose when applying a Δ-operator to a TBox $\mathcal{T}$.

Curiousities of the deductive closure

The deductive closure of $\Delta(\mathcal{T})$ may not contain all elements that the deductive closure of the original TBox $\mathcal{T}$ contained. To solve contradictions we require that at least all entailments of the form $U \sqsubseteq \bot \in (\mathcal{T})^\star$ are not be contained in $(\Delta(\mathcal{T}))^\star$. Definition 10 enables us to compare different operators by the entailments that are no more contained in the deductive closure of $(\Delta(\mathcal{T}))$.

With the definition of invalidating entailments in hand, we define an operator Δ on TBoxes such that the resulting structure meets the properties we defined in Section 1. We hence require this operator to meet certain conditions:

1. An entailment relation $\models$ is needed for the range of Δ.
2. The expressive power of the range of Δ must at least be as high as the expressive power of $\mathcal{T}$.
3. For a consistent Δ-Operator, its application to any $\mathcal{T}$ must not entail any concept U in the signature of $\mathcal{T}$ unsatisfiable.

While the first properties define a deductive closure for $\Delta(\mathcal{T})$, only the third property causes Δ to actually invalidate entailments. In particular, no axiom of the form $U \sqsubseteq \bot$ is contained in $(\Delta(\mathcal{T}))^\star$ for concepts U that are contained in the signature of $\mathcal{T}$. Nevertheless, we still allow for unsatisfiability for concepts. In particular for such unsatisfiable concepts as $U = A \sqcap \neg A$ — as long as U is not contained in the signature of $\mathcal{T}$.

Properties of the new operator

Note that Δ can invalidate other entailments as well. While this seems a flaw at the first sight, we will see in Section 6.2 that we have to indeed invalidate certain entailments which are not of the type $\Delta(\mathcal{T}) \models U \sqsubseteq \bot$ to obtain meaningful results.

Definition

The first two requirements define what a Δ-Operator is:

Definition 11 (Δ-Operator)**.** *Let $\mathcal{T}$ be a TBox. The operator Δ maps a TBox to the pair $\langle \Delta(\mathcal{T}); \models \rangle$ such that $\models$ defines an entailment relation for axioms from $\mathcal{L} = \mathbb{L}(\mathcal{T})$, i.e. the DL of $\mathcal{T}$.*

Note that we did not specify what $\Delta(\mathcal{T})$ actually is, just how it should behave. We have to be able to define an entailment relation $\models$ for $\Delta(\mathcal{T})$, and ensure that $\Delta(\mathcal{T})$ does not increase the expressive power compared to $\mathcal{T}$.

Coherency

The definition of Δ meets the first two requirements. In order to formalize the third requirement, i.e. that the entailment relation $\models$ does not entail any concept unsatisfiable, we introduce the notion of coherency for the pair $\langle \Delta; \models \rangle$:

Definition 12 (Coherency of a Δ-Operator)**.** *Let $\mathcal{T}$ be a TBox. A Δ-Operator is called coherent, if for any U that is in the signature of $\mathcal{T}$ it holds that $U \sqsubseteq \bot \notin (\Delta(\mathcal{T}))^\star$.*

3.2 The Null-Δ-Operator

The simple-most solution to solving conflicts is not to draw any logical consequence at all. The deductive closure of the result is simply the TBox itself.[1]

$$(\Delta(\mathcal{T}))^{\star} = \mathcal{T}$$

Entailment is defined by checking whether an entailment is actually contained in the TBox:

$$\Delta(\mathcal{T}) \models \eta \Leftrightarrow \eta \in \mathcal{T}$$

However, we can easily show that the Null-Δ-Operator is coherent.

Theorem 1. *The Null-Δ-Operator maps a TBox to the pair $\langle \mathcal{T}, \models \rangle$ with $\models (\eta)$, if and only if $\eta \in \mathcal{T}$. The Null-Δ-Operator is coherent.*

Proof. The conditions for being a Δ-Operator are trivially satisfied. Since axioms of the form $U \quad \sqsubseteq \quad \bot$ are not allowed to be contained in a TBox by definition, the Null-Δ-Operator is coherent. □

The Null-Δ-Operator is introduced for theoretical considerations only. In the sense of the conservation of implicit information property (P5 in Proposition 5) the Null-Δ-Operator is the worst Δ-Operator possible, because we loose all entailments of the original TBox $\mathcal{T}$. Please note that this operator indeed fulfills all other properties for a repair operator.

[1] In this special case, the deductive closure is finite.

3.3 The Repair Δ-Operators

Conflicts in a TBox $\mathcal{T}$ can be resolved by changing the TBox' axioms. This operation is usually referred to as *Ontology Repair*. In the case these operations only involve removing axioms a minimal repair is called *diagnosis*. Ontology Repair is the most commonly used technique for resolving unsatisfiable concepts in a TBox $\mathcal{T}$. In this section we show how Ontology Repair can be defined in terms of the Δ-Operator.

3.3.1 The Diagnosis-Δ-Operator

In order to resolve a conflict in a TBox, the according operation should change $\mathcal{T}$ as well as $(\mathcal{T})^+$ as little as possible. According to Reiter [1987b] a *diagnosis* for a set of entities $\mathcal{T}$ that contain a conflict is a minimal subset $\mathbb{D} \subseteq \mathcal{T}$ such that $\mathcal{T} \setminus \mathbb{D}$ does not contain a conflict anymore.[2] Minimal here means that there is no subset $\mathbb{D}'$ of $\mathbb{D}$ whose removal from $\mathcal{T}$ would result in solving the contradictions. In the case of unsatisfiable concepts for a TBox, we can reformulate this definition as follows:

Diagnosis and repair

Definition 13 (Diagnosis)**.** *Let $\mathcal{T}$ be a TBox and $U_0, \ldots, U_M$ be a set of concepts such that $\mathcal{T} \models U_m \sqsubseteq \bot$ for $0 \leq m \leq M$. A* diagnosis *$\mathbb{D}$ for $\mathcal{T}$ is a set of axioms from $\mathcal{T}$ such that (i) $\mathcal{T} \setminus \mathbb{D} \not\models U \sqsubseteq \bot$ for any (complex) concept in $sig(\mathcal{T})$ and (ii) for each $\mathbb{D}'$ that fulfills (i) we have that $\mathbb{D}' \not\subseteq \mathbb{D}$.*

In that sense, a diagnosis is the dual of the set of all root JUCs. Reiter [1987b] proposed to compute diagnoses using a Hitting Set

[2] The term diagnosis is sometimes referred to as repair plan. Diagnoses must not be mixed up with root JUCs. Finding diagnoses is the dual problem of finding root JUCs.

Tree (HST)[3], which can be reformulated as computing the dual HST for obtaining JUCs.

Using diagnoses, we can define a Δ-Operator that works on diagnoses and provides a coherent Multi-View TBox.

Definition 14. *Let $\mathcal{T}$ be a TBox and $\mathbb{D}$ be a diagnosis for $\mathcal{T}$. The mapping $\Delta(\mathcal{T}) = \mathcal{T} \setminus \mathbb{D}$ together with the classical entailment relation $\models$ defines the Diagnosis-Δ-Operator.*

Theorem 2. *The Diagnosis-Δ-Operator from Definition 14 is coherent.*

Proof. If we remove axioms from a TBox and/or an RBox, then the expressive power of the corresponding subsets can obviously not increase:

Lemma 1. *Let $(\mathcal{T}, \mathcal{R})$ be a TBox and RBox[4]. For any subset $\mathcal{T}' \subseteq \mathcal{T}$, $\mathcal{R}' \subseteq \mathcal{R}$ it holds that*
$\mathbb{L}(\mathcal{T}, \mathcal{R}) \geq \mathbb{L}(\mathcal{T}', \mathcal{R}')$

According to Lemma 1, $\mathcal{T} \setminus \mathbb{D}$ is expressible in the same DL $\mathcal{L}$ as was $\mathcal{T}$. By definition, an entailment relation $\models$ is defined on the range of Δ. Hence, the Diagnosis-Δ-Operator fulfills all conditions required for a Δ-Operator specified in Definition 11. Since, by definition, $\mathcal{T} \setminus \mathbb{D}$ does not entail any concept in the signature of $\mathcal{T} \setminus \mathbb{D}$ unsatisfiable, the Diagnosis-Δ-Operator is coherent. □

Any repair Δ-Operator involves a change in the original knowledge base and hence violates the originality property. Instead of removing an explicitly stated axiom, we propose to split up the TBox such that groups of axioms causing unsatisfiability are not

[3] It should be noted that in the original version of Reiter [1987b], diagnoses were defined on Hitting-Set-Trees. However, although not explicitly stated in the literature, diagnoses—and therefore also justifications—have to be computed on *Minimal* HSTs.

[4] Note that we restrict ourselves to the case of TBox axioms. We mention the RBox here just for the sake of completeness.

used at the same time when drawing logical consequences. That way, we can resolve unsatisfiable concepts. We keep the formalism for knowledge representation but have to slightly change the entailment relation. These procedures are described in detail in Chapter 4.

3.3.2 The All-Repair-Δ-Operator

The first approach developed in this thesis

For a TBox with unsatisfiable concepts, there can exist more than one diagnosis. In the best case, all root JUCs share one axiom, which results in a single diagnosis. In the worst case, all axioms in all root JUCs are pairwise distinct. We then have $|J^0| \cdot \ldots \cdot |J^k|$ different diagnosis. If N denotes the maximal cardinality of a root JUC, we have at most N^k different diagnoses in the worst-case. The choice of the diagnosis is non-deterministic, as a result. We can define a deterministic Repair-Δ-Operator by setting $\Delta_{\mathcal{T}} = \mathcal{T} \setminus \mathcal{J}$ where $\mathcal{J}$ is the union of the set of all root JUCs.

Definition 15 (All-Repair-Δ-Operator)**.** *Let $\mathcal{T}$ be a TBox and $\mathcal{J}$ be the set of all root JUCs w.r.t. $\mathcal{T}$. The All-Repair-Δ-Operator is defined as the pair $\langle \mathcal{T} \setminus (\bigcup_{k=0}^{K} J^k_{\mathcal{T} \models U \sqsubseteq \bot}); \models \rangle$ with the classical entailment relation $\models$.*

This repair operator is coherent:

Theorem 3. *The All-Repair-Δ-Operator from Definition 15 is coherent.*

Proof. Obviously $\mathcal{J} \supset \mathbb{D}$ for any diagnosis for $\mathcal{T}$. The Diagnosis-Δ-Operator is coherent. Since we learned that $\models$ is monotone, removing more axioms cannot re-introduce entailments the form $\Delta(\mathcal{T}) \models U \sqsubseteq \bot$ for any U in the signature of $\Delta(\mathcal{T})$. The All-Repair-Δ-Operator is a coherent Δ-Operator, as a consequence. □

Applying the All-Repair-Δ-Operator we invalidate more entailments than using the Diagnosis-Δ-Operator. On the other hand, finding a solution is deterministic. We will see the same trade-off between deterministic and minimal invasive Δ-Operators in Chapter 4 where we investigate Δ-Operators based on Default Logic.

3.4 A $\triangle$-operator for Multi-View Knowledge Bases

The second approach developed in this thesis

To invalidate the entailments $\mathcal{T} \models U \sqsubseteq \bot$ we must not consider all axioms of the root JUCs at the same time when drawing logical consequences. While we saw how this could be achieved for ontology repair, we now present a method where *we do not have to remove any axioms* to obtain a coherent Δ-Operator.

3.4.1 Multi-View Knowledge Bases

We *split up parts of the root JUCs* into non-empty sets $\mathcal{U}_0, \ldots, U_N$ and a separate (possibly empty) TBox $\mathcal{T}_\triangle$. The $\mathcal{U}_n$, $n = 0, \ldots, N$ contain axioms from the root JUCs such that for each root JUC two different axioms are contained in two different $\mathcal{U}_n$. What axioms actually are contained in the $\mathcal{U}_n$ will be subject to Chapter 5. In the separate TBox $\mathcal{T}_\triangle$ we collect the axioms that are not part of a conflict, i.e. $\mathcal{T} \setminus \bigcup_{k=0}^{K} J^k_{U \sqsubseteq \bot}$, as well as those axioms from the root JUCs that are not added to any $\mathcal{U}_n$.

The separate TBox $\mathcal{T}_\triangle$ models a conceptualization that is always valid whereas the sets $\mathcal{U}_n$ model certain *aspects* of the world. We therefore call $\mathcal{T}_\triangle$ *the Universal TBox* and the TBoxes $\mathcal{U}_n$ *Aspect TBoxes*. We require both of them to be coherent. Together with $\mathcal{T}_\triangle$ every Aspect TBox defines a coherent *view on the world*, whereas coherency is not guaranteed for $\mathcal{T}_\triangle \cup \mathcal{U}_n \cup \mathcal{U}_m$ for $U_n \neq U_m$. We collect all the information in all views $\mathcal{T}_\triangle \cup \mathcal{U}_n$ in a single *Multi-View-TBox*:

Definition 16. *Let $\mathcal{T}_\Delta$ and $\mathcal{U}_0, \ldots, \mathcal{U}_N$ be disjoint TBoxes. A Multi-View-TBox $\mathcal{DT}$ is the finite family of non-empty TBoxes:*

$$\mathcal{DT} = (\mathcal{T}_\Delta \cup \mathcal{U}_0, \ldots, \mathcal{T}_\Delta \cup \mathcal{U}_N)$$

3.4.2 Entailment Relation for a Multi-View-TBox

Since a Multi-View-TBox is a wrapper for a set of regular TBoxes, inference services like checking axiom satisfiability and consistency can be applied by means of the inference service defined for the original TBox. We hence use the classical DL entailment relation on the union of each Aspect TBox with the Universal TBox, but consider each of these unions separately. Consequently, we introduce the following definitions:

A Multi-View-TBox $\mathcal{DT}$ is said to entail an axiom, $\mathcal{DT} \models \eta$, if one of its TBoxes entails η. As a result, the deductive closure of a Multi-View-TBox $\mathcal{DT}$ is defined by $(\mathcal{DT})^\star = \bigcup_{n=0}^{N} (\mathcal{T}_n)^+$. We say that $\mathcal{DT}$ is satisfiable if all of its TBoxes are satisfiable.

However, because we did not remove any axioms, the contradiction is still in the data w.r.t. the classical entailment relation. The entailment relation for a Multi-View-TBox just hides that by not entailing concepts that occur in the signature of $\mathcal{T}$ unsatisfiable.

3.4.3 Invalidated Entailments

As we consider axioms separately for drawing logical consequences, we will invalidate entailments. In the sequel, we show which entailments we invalidate by separating axioms into different partitions,

Since partitions are mutually disjoint, the entailments that two different Aspect TBoxes $\mathcal{T}_n, \mathcal{T}_m$ of the resulting Multi-View-TBox have in common is at least the deductive closure of $\mathcal{T}_\Delta$:

$$(\mathcal{T}_n)^\star \ \cap \ (\mathcal{T}_m)^\star \supseteq (\mathcal{T}_\Delta)^\star$$

The intersection of the deductive closures of two single Aspect TBoxes, $(\mathcal{T}_n)^\star \ \cap \ (\mathcal{T}_m)^\star$, may indeed contain more entailments than $(\mathcal{T}_\Delta)^\star$. This is illustrated by the following example:

Example 8. *Consider the Multi-View TBox*

$$\mathcal{T}_0 = \{\top \sqsubseteq \forall R.A, A \sqsubseteq C\}, \mathcal{T}_1 = \{\top \sqsubseteq \forall R.B, B \sqsubseteq C\}, \mathcal{T}_\Delta = \emptyset$$

Each $\mathcal{T}_{\{0,1\}}$ entails $\top \sqsubseteq \forall R.C$, but $\mathcal{T}_\Delta$ obviously not.

Compared to the original TBox, in the Multi-View-TBox all entailments are invalidated whose justifications share axioms that are contained in different partitions:

$$(\mathcal{T})^\star \setminus (\mathcal{DT})^\star = \{\, \eta \,|\, J_{\mathcal{T},\eta} \cap \mathcal{U}_n \neq \emptyset \ \wedge \ J_{\mathcal{T},\eta} \cap \mathcal{U}_m \neq \emptyset \ \wedge \ n \neq m\}$$

In Table 3.1 the invalidated entailments are shown for three different possible Multi-View-TBox for the TBox of Example 2. In all cases, the unwanted entailments $C \sqsubseteq \bot$ and $D \sqsubseteq \bot$ are invalidated. For the first two Default TBoxes, the entailments $C \sqsubseteq A$ and $D \sqsubseteq A$ are not valid anymore whereas these entailments are preserved in the third case.
In the first two cases, the two axioms $B \sqsubseteq A$ and $C \sqsubseteq B$ are in different partitions. Yet, the only justification for the entailment $C \sqsubseteq A$ consists of exactly these two axioms. Because we do consider them separately for drawing logical consequences, the entailment $C \sqsubseteq A$ is not valid in the first two Multi-View-TBox. In the third case, in contrast, we have that $\{B \sqsubseteq A, C \sqsubseteq B\} \subset \mathcal{T}_\Delta \cup \mathcal{U}_0$. Hence, the entailment $C \sqsubseteq A$ is still valid for the third Multi-View-TBox.

If we applied classical ontology repair, i.e. axiom removal, not only we delete the removed axioms from the deductive closure of the TBox, but even entailments whose justifications contain just one of the removed axiom are invalidated. Using our method,

	$\mathcal{DT}$			$(\mathcal{T})^+ \setminus (\mathcal{DT})^+$
	$\mathcal{T}_\Delta$	$\mathcal{U}_0$	$\mathcal{U}_1$	
1	$D \sqsubseteq C$	$B \sqsubseteq A$	$C \sqsubseteq B$ $C \sqsubseteq \neg A$	$C \sqsubseteq A$ $D \sqsubseteq A$
2	$C \sqsubseteq \neg A$ $D \sqsubseteq C$	$B \sqsubseteq A$	$C \sqsubseteq B$	$C \sqsubseteq A$ $D \sqsubseteq A$
3	$C \sqsubseteq B$ $D \sqsubseteq C$	$B \sqsubseteq A$	$C \sqsubseteq \neg A$	

Table 3.1: All possible coherent Multi-View-TBoxes $\mathcal{DT}$ with $\mathcal{DT} = (\mathcal{T}_\Delta, \mathcal{U}_0, \mathcal{U}_1)$ for the TBox of Example 2 and the set of invalidated entailments $(\mathcal{T})^+ \setminus (\mathcal{DT})^+$.

in contrast, we only invalidate justifications that contain pairs of axioms—both of which are contained in different partitions. We hence claim that using our method for defining the Δ-operator, fewer entailments are subject to invalidation and more knowledge is preserved.

Repair is more invasive than separation.

For Example 2 all possible minimal repair solutions are shown in Table 3.2. In the first two cases, we remove one axiom that is part of the single existing justification for $C \sqsubseteq A$, which causes this entailment to become invalid. In the best case, we only lose the removed axiom itself. This happens when the removed axiom is not part of any justification for a non-trivial entailment.

3.4.4 A Δ-operator for Multi-View Knowledge Bases

We can define a Δ-operator in such a way that it produces a coherent Multi-View-TBox. We therefore define the aspect TBoxes $\mathcal{U}_n$ such that (i) for each $J^k_{U \sqsubseteq \bot}$ at least two axioms are contained

	Removed Axiom	$(\mathcal{T})^{+} \setminus (\mathcal{DT})^{+}$
1	$B \sqsubseteq A$	$B \sqsubseteq A$ $C \sqsubseteq A$ $D \sqsubseteq A$
2	$C \sqsubseteq B$	$C \sqsubseteq B$ $C \sqsubseteq A$ $D \sqsubseteq A$
3	$C \sqsubseteq \neg A$	$C \sqsubseteq \neg A$

Table 3.2: All possible ontology repair solutions for the TBox of Example 2 and the set of invalidated entailments $(\mathcal{T})^{\star} \setminus (\mathcal{DT})^{\star}$.

in two different $\mathcal{U}_n$ and $\mathcal{U}_m$ and (ii) no root JUC is contained in any of these aspect TBoxes.

Definition 17. *Let $\mathcal{T}$ be a TBox with unsatisfiable concepts and $J^k_{U \sqsubseteq \bot}$ with $k = 0, \ldots, K$ the root JUCs for them. The Splitting-Δ-operator defines a finite family of subsets $(\mathcal{T}_{\Delta} \cup \mathcal{U}_n)_{n=0}^{N}$ of $\mathcal{T}$ such that for each root JUC $J^k_{U \sqsubseteq \bot}$:*

- *(i) there exist $\alpha, \beta \in J^k_{U \sqsubseteq \bot}$ such that $\alpha \neq \beta$, $\alpha \in \mathcal{U}_n$, $\beta \in \mathcal{U}_m$ and $m \neq n$.*
- *(ii) it holds that $J^k_{U \sqsubseteq \bot} \not\subseteq \mathcal{T}_{\Delta} \cup \mathcal{U}_n$ for all $n = 0, \ldots, N$.*

Condition (i) ensures that indeed all unsatisfiable concepts from the original TBox are resolved and (ii) ensures that none of the original justifications is re-introduced through the back-door. As a result, we obtain the following theorem:

Theorem 4. *The Multi-View-TBox resulting from the Splitting-Δ-Operator is coherent.*

This is an interesting result, but the question remains whether such a Splitting-Δ-operator exists. Fortunately, we can show that, under certain conditions, such an operator indeed exists. These conditions, in turn, appear to be nothing special: We require the justifications to be free of cycles and prohibit explicit contradictions. We can then split up each root justification into two sets such that we may put one axiom from each set into one of two aspect TBoxes. This splitting, which is described in detail in Chapter 4, defines a Splitting-Δ-operator. Different variants of the Splitting-Δ-operator are investigated in Chapter 5.

Coherency of this operator

3.5 Belief Revision

Belief revision (also referred to as belief change) describes the process of whether and how to change the current beliefs of an agent to ensure that his beliefs are consistent with the current state of the world. This is similar to what we defined the Δ-Operator for. Belief revision is a frequently used concept in artificial intelligence, and therefore we will show how we can express the Δ-Operators, that we introduced in the previous sections, within the context of belief revision.

3.5.1 The AGM Postulates

Belief revision was given a formal definition by the AGM postulates [Alchourron et al., 1985]. This framework, which was revised itself several times, studies idealized mathematical models of belief revision. For a given logic language $\mathcal{L}$, the beliefs of an agent are represented by a (possibly infinite) set of axioms in $\mathcal{L}$. This set, denoted by $\mathcal{B}$, is referred to as the *belief set* and is closed under logical consequence.[5] New evidence Ω are sets of axioms

Formalizing belief revision

[5] This requires that a logical consequence operator is defined. Furthermore, $\mathcal{B}$ is a belief set if and only if $\mathcal{B} = (\mathcal{B})^\star$.

in $\mathcal{L}$, too, and to incorporate this new evidence, a *belief revision operator* is defined such that a new belief set can be obtained from the current belief set.

Belief revision can be roughly divided into three groups: *contraction*, i.e. axioms are removed from $\mathcal{B}$, *expansion*, i.e. axioms are added to $\mathcal{B}$, and *revision* which is a combination of the former two. Applying one of these operators to a belief set is denoted by $\mathcal{B} \circ \Omega$. In any case, the result is required to be consistent.

The outcome of contracting a belief set $\mathcal{B}$ by new evidence $\Omega = \{\omega_0, \ldots, \omega_P\}$ should be a subset of $\mathcal{B}$. This subset must not entail Ω, but should be minimal w.r.t. $\mathcal{B}$ in order to not remove unnecessarily many axioms from K. Consequently, it is of interest to consider the inclusion-maximal subsets of $\mathcal{B}$ that do not entail Ω.

Definition 18 (AGM Postulates [Alchourron et al., 1985])**.**
Let $\mathcal{B}$ be a belief set and $\mathcal{B} \circ \Omega$ be a belief revision. The AGM postulates are defined as follows:

AGM1 (Closure): $\mathcal{B} \circ \Omega = (\mathcal{B} \circ \Omega)^\star$

AGM2 (Success): *If* Ω *is inconsistent, then* $\Omega \not\subseteq (\mathcal{B} \circ \Omega)^\star$

AGM3 (Inclusion): $\mathcal{B} \circ \Omega \subseteq \mathcal{B}$

AGM4 (Vacuity): *If* $\Omega \not\subseteq (\mathcal{B})^\star$*, then* $\mathcal{B} \circ \Omega = \mathcal{B}$.

AGM5 (Preservation): *If* $(\Omega_1)^\star = (\Omega_2)^\star$*, then* $\mathcal{B} \circ \Omega_1 = \mathcal{B} \circ \Omega_2$.

AGM6 (Recovery): $\mathcal{B} \subseteq ((\mathcal{B} \circ \Omega) \cup \Omega)^\star$.

3.5.2 Δ-Operators as Contraction Operators

Considered as contraction operators, all the Δ-Operators presented in the previous sections of this chapter fulfill all AGM postulates. To see this, we define the set of current beliefs $\mathcal{B}$ as the deductive closure of a TBox $\mathcal{T}$. We denote the entailments that the Δ-Operator removes from $(\mathcal{T})^\star$ by Ω. In our case $\mathcal{B} \circ \Omega$ means $(\mathcal{T})^\star \setminus \Omega$ with $\Omega \subseteq (\mathcal{T})^\star$. In other words: $(\mathcal{B} \circ \Omega)^\star = (\Delta(\mathcal{T}))^\star$. Note

How the Δ-Operators relate to belief revision.

that the AGM postulates, in general, are defined on the infinite deductive closure.

Since we remove entailments from the deductive closure of $\mathcal{T}$, we fulfill AGM1, AGM3 and AGM6 by definition. We also fulfill AGM4, since $\Omega \subset (\mathcal{T})^\star$. As we perform axiom removal, Ω is not included in $((\mathcal{T}) \circ \Omega)^\star$, so we fulfill AGM2.

For the fulfillment of AGM5 we consider two arbitrary subsets of $(\mathcal{T})^\star$ whose deductive closures do not differ, i.e. $(\Omega_1)^\star = (\Omega_2)^\star$. Assume further $(\mathcal{T})^\star \setminus \Omega_1 \neq (\mathcal{T})^\star \setminus \Omega_2$. Note that $\Omega_{\{1,2\}} \subseteq (\mathcal{T})^\star$, which requires the existence of ω such that $\omega \in \Omega_1$, but $\omega \notin \Omega_2$ (or vice versa). The deductive closure of Ω_2 is defined according to classical entailment. Consequently, we have that $\omega \notin (\Omega_2)^\star$. From the prerequisites of AGM5 we further conclude that $\omega \notin (\Omega_1)^\star$. Finally, we have that $\omega \notin \Omega_1$ which contradicts our assumption. Hence AGM5 is fulfilled, too.

If we require the deductive closure $(\Delta(\mathcal{T}))^\star$ to be a belief set itself, the logical consequence operator must be defined according to that of the Δ-Operator. For all the Δ-Operators considered in this thesis, we can safely do so, because their logical consequence operators are based on the classical one.

Note that not all Δ-Operators are also contraction operators that fulfill the AGM postulates. This is, for example, the case for axiom rewriting, which is indeed a belief revision operator. The question whether or not requiring a Δ-Operator to fulfill the AGM postulates is left open for discussion. More information on belief revision can be found in Gärdenfors [2003].

4

Default Logics for Solving Conflicts

Default Logics can be used for solving conflicts in a Description Logics TBox. Under certain restrictions, we can guarantee a consistent Multi-View TBox whose Aspect TBoxes are constructed according to the partitioning algorithm provided by Lehmann's Default Logics (cf Section 2.5). If we already computed all JUC for the unsatisfiable concepts of a TBox we can define a splitting scheme that provides these partitions without the need to perform additional reasoning. Based on this splitting scheme we define a general algorithm that serves as the base for all Splitting-Δ-Operators, which are defined in Chapter 5.

Plan of This Chapter

We start this chapter with showing how we use Default Logics for solving conflicts in Section 4.1. In the subsequent Section 4.2 we provide the restrictions under which our approach works. The splitting that avoids additional reasoning is introduced in Section 4.3 and is the basis for the transformation algorithm in Section 4.4.

4.1 Default Logics for Solving Conflicts

As we saw in Section 2.5, Default Logics allow to draw logical consequences in the presence of controversial information. Using Probabilistic Description Logics we may, furthermore, split up our knowledge base into a crisp and a default part. We will show in this section, how we can facilitate the Default Logics procedure for finding partitions to find a coherent Multi-View TBox.

4.1.1 Default TBox

The partitioning algorithm of (Probabilistic) Default Logics [Lukasiewicz, 2008] splits up the sets of default formulas $\mathcal{D}$ into partitions $\mathcal{U}_0, \ldots, \mathcal{U}_N$ such that none of these partitions entails a concept unsatisfiable. Considering these formulas to be a set of TBox axioms, the set of TBox axioms is split up in such a way that none of the sets $\mathcal{U}_n$ (which are sets of axioms and, consequently, TBoxes) entails any concept unsatisfiable—as long as the last remainder set D_N is empty (cf Section 2.5). Hence, Default Logics on TBox axioms induces a Multi-View TBox where the Aspect TBoxes happen to be the partitions from Default Logics.

Default Logics for solving unsatisfiable concepts

We now put the cart before the horse and show that for a given valid partitioning of a Default Knowledge Base, solely consisting of TBox and RBox axioms, we can extend all partitions by certain axioms and not affect satisfiability. Let therefore $\mathcal{U}_0, \ldots, \mathcal{U}_n$ be a partitioning of a Default Knowledge Base that was computed according to Lukasiewicz [2008]. According to monotonicity, we may extend each of the partitions $\mathcal{U}_n$ by additional axioms $\alpha_i \in \mathcal{T}$ for $i = 0, \ldots, I$ as long as this does not cause to entail a concept in the signature of $\mathcal{U}_n \cup \{\alpha_0, \ldots, \alpha_I\}$ unsatisfiable. We collect $\alpha_0, \ldots, \alpha_I$ in a separate TBox $\mathcal{T}_\Delta$ and come up with a coherent Multi-View TBox.

Theorem 5. *Let $\mathcal{T}_\Delta$ be a TBox and $\mathcal{U}_0, \ldots, \mathcal{U}_N$ the partition of a Default Knowledge Base that solely consists of TBox axioms. If $\mathcal{T}_\Delta \cup \mathcal{U}_n \not\models U \sqsubseteq \bot$ for any n, U, then $(\mathcal{T}_\Delta \cup \mathcal{U}_0, \ldots, \mathcal{T}_\Delta \cup \mathcal{U}_N)$ is a coherent Multi-View TBox.*

We refer to Multi-View TBoxes whose Aspect TBoxes fulfill the partition properties of Default Logics as *Default TBoxes*. This allows to distinguish them from such Multi-View TBoxes for which we are not able to apply Default Logics entailment.

4.1.2 Solving Conflicts

We now show how Default Logics can be applied to our original problem of drawing logical consequences for an incoherent TBox $\mathcal{T}$. The only axioms that are involved in a conflict are those axioms of the root JUC for $\mathcal{T}$. Hence, we only have to declare these axioms as Defaults, whereas we may put all other axioms into the Universal TBox $\mathcal{T}_\Delta$. We will show later on that, under certain restrictions, the resulting Default TBox is coherent (cf Section 5.1.1).

Checking the fulfillment of the properties

Coherency is one of the key requirements we defined in Section 1, but what about the other properties? We did not change- the knowledge representation which fulfills property P1. The partitions and $\mathcal{T}_\Delta$ can be computed solely from the TBox itself by computing justifications and applying the partitioning procedure defined in Lehmann [1995]. This meets property P3 which requires the procedure to work automatically. By definition, the partitions $\mathcal{U}_n$ together with $\mathcal{T}_\Delta$ partition the *original* TBox, i.e. $\mathcal{T} = \mathcal{T}_\Delta \oplus \mathcal{U}_0 \oplus \ldots \oplus \mathcal{U}_N$. Hence, we also fulfill property P4. Hence, Default Logics allows us to define a Δ-Operator such that the partitioning of a TBox is transformed into a *Default TBox*. We will show we this can be accomplished efficiently in Section 5.1.1.

Please note that while we identify Default TBoxes with Multi-View-TBoxes, the entailment relation for a Default TBox is not the same as defined by Lehmann's Lexicographical Entailment: We borrowed the method of how to separate axioms from each other into partitions but we compute regular models instead of lex-mini-mal models. While it was possible to compute lex-minimal models, we prefer the classical entailment on the single Views of Multi-View-TBoxes: On the one hand, we just want to invalidate entailments, on the other hand, we do not want to introduce the full complexity of Lehmann's Default Logics.

4.2 Restrictions on the TBox

The main idea behind resolving conflicts by treating trouble-causing axioms separately is to *split up the set of root JUC*. We can do this in a way that for each root JUC those axioms with the unsatisfiable concept on the left-hand-side are collected in one set, Γ. All other axioms of this root justification are collected in a separate set Θ. Both sets are called *splitting sets* and both sets partition each root JUC [Scharrenbach et al., 2010c].

Under certain conditions, we can show that neither of the splitting sets is empty. The non-emptiness of the splitting-sets is one of the crucial conditions for all Splitting-Δ-Operators that define a consistent Multi-View-TBox. The splitting not only allows to define Splitting-Δ-Operators, we may compute them *without additional satisfiability checks*.

In this section we show what restrictions have to be made on the TBox such that the partitioning scheme produces a consistent Multi-View TBox. We remember Definition 8 (TBox contradictions) which identifies contradictions in a TBox with unsatisfiable concepts of that TBox. The restrictions on the TBox being made in the following exclude certain types of TBox contradictions that must not occur in a TBox. Along with the definition of those forbidden contradictions, we provide ways how to solve them. We therefore have to apply certain changes to the TBox that transform those forbidden contradictions into admissible contradictions, i.e. such contradictions that the method we present in this work is able to solve.

4.2.1 Logical Contradictions

Logical contradictions are axioms that contain the conjunction $A \sqcap \neg A$ on the right-hand side of an axiom or explicitly state $U \sqsubseteq \bot$ in the TBox. We will give a detailed description how to handle either case.

Explicitly Stated Logical Contradictions

Contradictions that are caused by the latter case cannot be solved except by removing the corresponding axioms or parts thereof from the TBox. But even detecting axioms of the latter case is not as obvious as it seems at a first glance. While we can easily detect and remove axioms of the form $U \sqsubseteq \bot$ or, equivalently, axioms of the form $\top \sqsubseteq \neg U$ things become difficult when a disjunction is involved. We illustrate this by the following example:

Example 9 (Explicitly stated logical contradictions**).**
Assume the TBox $\mathcal{T} = \{\top \sqsubseteq \neg U \sqcup (B \sqcap C), \neg U \sqcup (\neg B) \sqsubseteq \top\}$. *Obviously,* $\mathcal{T} \models U \sqsubseteq \bot$ *with the whole TBox as a justification. The interpretation of the left-hand-side of both axioms is equal to the whole interpretation domain.*
Hence $\Delta^{\mathcal{I}} \subseteq ((\Delta^{\mathcal{I}} \setminus U^{\mathcal{I}}) \cup (\Delta^{\mathcal{I}} \setminus B^{\mathcal{I}})) \cap ((\Delta^{\mathcal{I}} \setminus U^{\mathcal{I}}) \cup (B^{\mathcal{I}} \cap C^{\mathcal{I}}))$ *which implies that* $U^{\mathcal{I}} = \emptyset$.

In Example 9 the problem arises from the disjunct $\neg U$ on the right-hand-side and the top-concept on the left-hand-side of an axiom. We can apply DeMorgan-rules:

$$\top \sqsubseteq \neg\neg(\neg U \sqcup (B \sqcap C))$$

Pushing the negation inwards we can consider them as a disjoint.

$$\top \sqsubseteq \neg(U \sqcap \neg(B \sqcap C))$$

As we learned in Section 2.1, we can rewrite this disjoint without changing its semantics to

$$U \sqcap \neg(B \sqcap C) \sqsubseteq \bot$$

We hence explicitly stated that the complex concept $U \sqcap \neg(B \sqcap C)$ is unsatisfiable which we just classified as not admissible.

There exist several options how to solve this problem. First of all, we note that we do not have to prohibit axioms of the form $\top \sqsubseteq C$ in general. Trouble arises when the complex concept C contains a negated disjunct U that is unsatisfiable. We observe that $U^{\mathcal{I}} = \emptyset$, and consequently $(\neg U)^{\mathcal{I}} = \Delta^{\mathcal{I}}$. Hence the disjunction $\neg U \sqcup D$ cannot provide any meaningful information about D, since $(U \sqcup (B \sqcap C))^{\mathcal{I}} = \Delta^{\mathcal{I}}$ no matter what $(B \sqcap C)$ is. We therefore propose to remove the non-informative disjunct U from C for all occurrences of C in a root JUC.

Implicitly Stated Logical Contradictions

Since $A \sqcap \neg\, A \;\equiv\; \bot$ for any axiom of the form $U \;\sqsubseteq\; A \sqcap \neg\, A \sqcap B$ there exist two way of how to interpret the logical contradiction:

1. The concept U must never be instantiated, i.e. it represents a forbidden entity.

2. $A \;\sqcap\; \neg\, A$ is a modeling error.

In the scope of this thesis we consider only case 2, because we want the TBox not to entail any conflict. There exist several possible ways of how to resolve the logical contradiction when considered as modeling error:

1. Deletion: $A \sqcap \neg\, A$ is a modeling error, hence we completely delete it.

2. Partial deletion: We only delete one of the concepts, either A or $\neg\, A$ from the axiom.

3. Axiom rewriting: We split up the axiom into two axioms $U \;\sqsubseteq\; A \;\sqcap\; B$ and $U \;\sqsubseteq\; \neg\, A \;\sqcap\; B$.

Yet, none of these options is optimal in a general sense. The choice depends on the current situation. While solution 1 simply removes information, solution 2 keeps part of it. Applying solution 3, we have to rewrite the axiom, but we keep the information provided. Since *we consider all information originally provided as important*, we propose solution 3 within the scope of this thesis. However, solution 3 results in a direct contradiction which are described in the following section. In the sequel, we assume every TBox $\mathcal{T}$ is free of logical contradictions.

Original information is considered important.

The absence of logical contradictions is vital for solving conflicts by treating its conflicting axioms separately. If a TBox $\mathcal{T}$ does not contain axioms with logical contradictions, then every JUC w.r.t. $\mathcal{T}$ has at least two axioms:

Lemma 2. *Let $\mathcal{T}$ be a TBox none of whose axioms contains a logical contradiction. Then for all JUC $J_{\mathcal{T},U\sqsubseteq\bot}$ for $\mathcal{T}$ it hold that $|J_{\mathcal{T},U\sqsubseteq\bot}| \geq 2$.*

The proof of Lemma 2 can be found in Appendix B.1.

4.2.2 Direct Contradictions

Default Logics cannot solve conflicts that are caused by two axioms whose left-hand-sides are subsumed by concepts that are disjoint. It refers to *explicitly stating* that something is a subclass of a particular concept and its complement at the same time. For example, the TBox $\mathcal{T} = \{U \sqsubseteq A, U \sqsubseteq \neg A\}$ contains a direct contradiction whereas the TBox $\mathcal{T} = \{U \sqsubseteq B, B \sqsubseteq A, U \sqsubseteq \neg A\}$ does not. Both TBoxes entail $U \sqsubseteq A$ and $U \sqsubseteq \neg A$ (and, obviously, $U \sqsubseteq \bot$). The difference is that in the first case both *entailments are stated explicitly* while in the latter TBox *one of these entailments implicit*.

Explicitly stated contradictions

Consider another example, $\mathcal{T} = \{U \sqsubseteq (\leq 1R.\top), U \sqsubseteq (\geq 2R.\top)\}$. Obviously, the TBox entails the two concepts $(\leq 1R.\top)$ and $(\geq 2R.\top)$ to be disjoint, but one concept is not the negation of the other. Hence, direct contradictions require a more complicated notion of disjointness for two concepts.

In the presence of a direct contradiction, the disjointness of the two concepts involved must be an explicit logical consequence but not an implicit one. An explicit disjointness of two concepts A_1 and A_2 must result from the concepts themselves. In particular, it must not rely upon any axiom besides the declarations $A_1 \sqsubseteq \top$ and $A_2 \sqsubseteq \top$.[1] We may express this using the notion of justifications:

Definition 19. *Let $\mathcal{T}$ be a TBox. The concepts A_1 and A_2 are said to be* explicitly disjoint *w.r.t. $\mathcal{T}$, if $\mathcal{T} \models A_1 \sqcap A_2 \sqsubseteq \bot$ and $J_{\mathcal{T},A_1 \sqcap A_2 \sqsubseteq \bot} \setminus \{A_1 \sqsubseteq \top, A_2 \sqsubseteq \top\} = \emptyset$.*

Returning to Example 5, neither pair of concepts is explicitly disjoint. The only pair of axioms that are implicitly disjoint is $Penguin$ and $FlyingAnimal$. Since these axioms are explicitly disjoint, we have:

$$J_{\mathcal{T},Penguin \sqsubseteq \neg FlyingAnimal} = \{Penguin \sqsubseteq \neg FlyingAnimal\} \neq \emptyset$$

In contrast to that, the two concepts A and $\neg A$ are explicitly disjoint for the TBox $\mathcal{T} = \{B \sqsubseteq A, B \sqsubseteq \neg A\}$.

The notion of explicit disjointness enables us to define direct contradictions:

Definition 20. *Let $U \sqsubseteq A_1$ and $U \sqsubseteq A_2$ be two axioms of a TBox $\mathcal{T}$. We say that $\mathcal{T}$ contains a* direct contradiction *for U, if A_1 and A_2 are explicitly disjoint. We denote a direct contradiction for a concept U by $\mathcal{DC}_U = \{U \sqsubseteq A_1, U \sqsubseteq A_2\}$*

Default Logics cannot work with Direct Contradictions.

In case a TBox contains a direct contradiction, we cannot apply the Default Logics to resolve the conflict. Assume the direct contradiction $\mathcal{DC}_{U_0} = \{U_0 \sqsubseteq A_1, U_0 \sqsubseteq A_2\}$ in a TBox $\mathcal{T}$. One of

[1] Frankly, the declaration axioms do not have to be part of the justification. We consider them only in case they are explicitly declared in the ontology.

the justifications for $U_0 \sqsubseteq \bot$ is the direct contradiction itself, i.e. $J^{k_0}_{\mathcal{T},U_0 \sqsubseteq \bot} = \{U_0 \sqsubseteq A_1, U_0 \sqsubseteq A_2\}$.

Putting the two axioms $U_0 \sqsubseteq A_1$ and $U_0 \sqsubseteq A_2$ into separate partitions would solve the conflict but violate the definition of partitions (cf Section 2.5.4). For neither of the two axioms can we find a model that satisfies U_0 and A_1 or U_0 and A_2.

Even worse, the direct contradiction will be part of the last remainder set and cause the resulting Multi-View TBox to be incoherent. Assume we already computed the partitions $\mathcal{U}_0, \ldots, \mathcal{U}_n$, and still have to solve $J^{k_0}_{\mathcal{T},U_j \sqsubseteq \bot}$. In other words, we have that $\mathcal{T}_\Delta \setminus \bigcup_{m=0}^{n} \mathcal{U}_m \models U_j \sqsubseteq \bot$. Thanks to monotonicity, we know that also $\mathcal{T}_\Delta \setminus \bigcup_{m=0}^{n} \mathcal{U}_m \models U \sqcap F \sqsubseteq \bot$ hold for any concept F. Hence, none of the axioms in $J^{k_0}_{\mathcal{T},U \sqsubseteq \bot}$ is a candidate for $\mathcal{U}_{n+1}$. As a consequence, no partition $\mathcal{U}_m$ can contain any axiom of $J^{k_0}_{\mathcal{T},U \sqsubseteq \bot}$ for $m = n+1, \ldots, N$. Hence $J^{k_0}_{\mathcal{T},U \sqsubseteq \bot}$ is contained in the last remainder set D_N. As this causes D_N to be non-empty, the resulting Default TBox is not consistent.

Philosophical Excursion

The idea behind solving conflicts by Default Logics is that we allow to state exceptions from a general case. The Default Logics perspective on Example 5 assume that there there is an exception to the general rule that all birds are flying animals, i.e. penguins. Assume we stated that penguins are flying animals as well. There is no exception to a general rule anymore, because the two rules are equally general, i.e. on the same level of generality. We may solve this dilemma by separating the two rules when drawing logical conclusions, i.e. by putting them into two different Aspect TBoxes. Yet, this changes the semantics of these Aspect TBoxes. Direct Contradictions do not express exceptions but model perpendicular views on the world.

A way how to interpret Direct Contradictions

How to resolve Direct Contradictions Anyway

It seems that the existence of direct contradictions make the application of the partitioning algorithm impossible. Indeed, there is a way how to obtain a valid partition for a TBox $\mathcal{T}$—even if there exists a direct contradiction. Obviously, the two axioms of a direct contradiction must not appear in the same partition. In case A_1 and A_2 are both entailed satisfiable in the n-th partitioning step [2] then we may simply choose one of the axiom for $\mathcal{U}_n$ and the other for $\mathcal{U}_{n+1}$. For neither partition $\mathcal{U}_n$ the concept U is entailed unsatisfiable.

There exists a way how to still fulfill the definition of the partitioning algorithm. We therefore have to make one of the explicit axioms implicit. To achieve this we introduce a new concept $\tilde{U}$ that does not appear in $\mathcal{T}$ and replace one of the axioms of the direct contradiction, e.g. $U \sqsubseteq A_1$, by the set $\{U \sqsubseteq \tilde{U}, \tilde{U} \sqsubseteq A_1\}$. If we ensure always chosing $U \sqsubseteq \tilde{U}$ to be part of $\mathcal{U}_n$ and $U \sqsubseteq A_2$ to fall into partition $\mathcal{U}_{n+1}$ while placing $\tilde{U} \sqsubseteq A_1$ into $\mathcal{T}_\Delta$, then even the entailment $U \sqsubseteq A_1$ is preserved.

4.3 Unsatisfiability Splitting

For resolving a JUC, we must not consider all of its axioms at the same time for entailment. This condition is already fulfilled when we treat *only two of its axioms separately*. We therefore reuse the splitting scheme defined in Scharrenbach et al. [2010c]. Besides it allows proving the existence of a consistent Multi-View TBox, using the unsat splitting we can compute the Aspect TBoxes *without additional satisfiability checks*. All the hard work has already been done by computing all the JUC. The necessary information we need lies in the splitting. We partition each JUC $J^k_{\mathcal{T},U \sqsubseteq \bot}$ into the *splitting sets*:

[2] This is the case, if $\mathcal{T} \setminus \mathcal{U}_0 \cup \ldots \mathcal{U}_{n-1} \not\models A_1 \sqcup A_2 \sqsubseteq \bot$.

$$\begin{array}{rcl} \Gamma^k_{U \sqsubseteq \bot} & = & \{U \sqsubseteq A \in J^k_{\mathcal{T},U \sqsubseteq \bot}\} \\ \Theta^k_{U \sqsubseteq \bot} & = & J^k_{\mathcal{T},U \sqsubseteq \bot} \setminus \Gamma^k_U \end{array}$$

We refer to axioms from the first as γ-axioms whereas we call axiom from the latter θ-axioms. As we will show later on, the separation of a justification into Θ and Γ provides exactly the separation we seek for computing Aspect TBoxes: for each root JUC $J^k_{\mathcal{T},U \sqsubseteq \bot}$ one or all θ-axioms are placed into one Aspect TBox $\mathcal{U}_n$ and one or all γ-axioms are placed into a different Aspect TBox $\mathcal{U}_{n+1}$. The remaining axioms are added to $\mathcal{T}_\Delta$.

θ-axioms and γ-axioms

Splitting Sets are Non-Empty

To be able to separate an JUC by using axioms from its splitting sets, we demand these splitting sets to be non-empty. We can ensure this by requiring the TBox containing the unsatisfiable concepts to fulfill one of the constraints defined in Section 4.2.2: In the absence of direct contradictions, the justification for an unsatisfiable concept always consists of at least two axioms: one that has the unsatisfiable concept on the left-hand side and one that does not:

Lemma 3. *Let $\mathcal{T}$ be a TBox that contains unsatisfiable concepts. If none of the root JUC contains a direct contradiction, then for every root JUC the splitting sets are non-empty.*

The proof for Lemma 3 can be found in Appendix B.2.

4.3.1 Examples

We given an example to illustrate how JUC for a TBox are split up into Θ- and Γ-sets. Example 10 and Figure 4.1 how the splitting works.

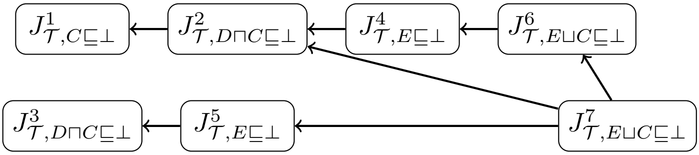

$J^1_{\mathcal{T},C\sqsubseteq\bot}$
$J^2_{\mathcal{T},D\sqcap C\sqsubseteq\bot}$
$J^4_{\mathcal{T},E\sqsubseteq\bot}$
$J^6_{\mathcal{T},E\sqcup C\sqsubseteq\bot}$
$J^3_{\mathcal{T},D\sqcap C\sqsubseteq\bot}$
$J^5_{\mathcal{T},E\sqsubseteq\bot}$
$J^7_{\mathcal{T},E\sqcup C\sqsubseteq\bot}$

Figure 4.1: Dependencies between the JUCs from Example 10. An arrow indicates that the justification at the start depends on the justifications at the end. Note that this dependency is a transitive relation.

Example 10 (Unsat-Splitting).
Assume the TBox $\mathcal{T}$ with the following axioms:

Specified				*Induced*			
τ_0 :	G	$\sqsubseteq$	A	τ_7 :	$D \sqcap C$	$\sqsubseteq$	$\top$
τ_1 :	B	$\sqsubseteq$	G	τ_8 :	$E \sqcup C$	$\sqsubseteq$	$\top$
τ_2 :	C	$\sqsubseteq$	B				
τ_3 :	C	$\sqsubseteq$	$\neg A$				
τ_4 :	D	$\sqsubseteq$	$\neg B$				
τ_5 :	E	$\sqsubseteq$	$D \sqcap C$				
τ_6 :	$E \sqcup C$	$\sqsubseteq$	$F \sqcap C$				
τ_9 :	$D \sqcap C$	$\sqsubseteq$	H				

The following concepts in the extended signature of $\mathcal{T}$ are unsatisfiable: C, $D \sqcap C$, E, $E \sqcup C$. The corresponding JUCs are:[3]

$$\begin{aligned}
J^1_{\mathcal{T},C\sqsubseteq\bot} &= \{\ \tau_0, \tau_1\ \} \cup \{\ \tau_2, \tau_3\ \}\\
J^2_{\mathcal{T},D\sqcap C\sqsubseteq\bot} &= \{\ \tau_0, \tau_1, \tau_2, \tau_3\ \} \cup \{\ \tau_7\ \}\\
J^3_{\mathcal{T},D\sqcap C\sqsubseteq\bot} &= \{\ \tau_2, \tau_4\ \} \cup \{\ \tau_7\ \}\\
J^4_{\mathcal{T},E\sqsubseteq\bot} &= \{\ \tau_0, \tau_1, \tau_2, \tau_3, \tau_7\ \} \cup \{\ \tau_5\ \}\\
J^5_{\mathcal{T},E\sqsubseteq\bot} &= \{\ \tau_2, \tau_4, \tau_7\ \} \cup \{\ \tau_5\ \}\\
J^6_{\mathcal{T},E\sqcup C\sqsubseteq\bot} &= \{\ \tau_0, \tau_1, \tau_2, \tau_3, \tau_5, \tau_7\ \} \cup \{\ \tau_8\ \}\\
J^7_{\mathcal{T},E\sqcup C\sqsubseteq\bot} &= \{\ \tau_0, \tau_1, \tau_2, \tau_3, \tau_4, \tau_5, \tau_7\ \} \cup \{\ \tau_8\ \}
\end{aligned}$$

[3] Note that we omitted the declarations for atomic concepts, since they do not play a vital role in the procedure.

The dependencies between these justifications are shown in Figure 4.1. We have that $J^1_{\mathcal{T},C\sqsubseteq\bot}$ and $J^3_{\mathcal{T},D\sqcap C\sqsubseteq\bot}$ are indeed root JUCs whereas the other JUCs are derived. It should be noted for both root JUCs the axiom τ_2 is contained in both a Γ-set (for $J^1_{\mathcal{T},C\sqsubseteq\bot}$) and in a Θ-set (for JUC $J^3_{\mathcal{T},D\sqcap C\sqsubseteq\bot}$). As we will see further on, this is the cause why we have three instead of only two partitions.

4.4 Algorithm

All different variants of the Splitting-Δ-Operators can be computed according to variants of Algorithm 1. The actual implementations differ in the choice of the axioms and the choice of the current partition where to add axioms to. In the following, we describe the generic algorithm for computing a consistent Multi-View TBox. Variations of this algorithm can be, for example, always choosing all axioms or choosing a single axiom only. They are described in detail in Chapter 5. We process all root JUCs until all of them are resolved. We collect the candidates for resolving in the current step in the sets Γ and Θ, and determine which of the JUCs are the root JUCs (Line 6). We store these in Θ_{root} and Γ_{root}. A JUC $J^k_{U\sqsubseteq\bot}$ is solved when its splitting sets $\Theta^k_{U\sqsubseteq\bot}$ and $\Gamma^k_{U\sqsubseteq\bot}$ are both empty.

How the Δ-Operator works.

We start the main iteration (Line 8) and process all root JUCs as long as not all their splitting sets have been resolved. The algorithm considers a splitting set as solved when it is empty. We first determine the Γ-sets (Line 12) that can be resolved. The finding task is described in detail in Section 4.4.4. Having resolved one or more Γ-sets, we resolve all dependent splitting sets (Line 17). Details on that can be found in Section 4.4.3. Afterwards, we add all those axioms that contain unsatisfiable concepts in the extended signature to the corresponding partitions (Line 43). How to perform this is subject to Section 4.4.6. Finally, we resolve axioms from Θ sets (Line 26), in particular those that are no more

Algorithm 1 Compute a consistent Multi-View TBox for a Splitting-Δ-Operator.

Function:

$\mathcal{DT} \leftarrow$ $computeMultiViewTBox(\mathcal{T}, \Theta^0_{U \sqsubseteq \bot}, \Gamma^0_{U \sqsubseteq \bot} \ldots, \Theta^K_{U \sqsubseteq \bot}, \Gamma^K_{U \sqsubseteq \bot})$

```
test
// Initialization
$\Theta \leftarrow \emptyset, \qquad \Gamma \leftarrow \emptyset, \qquad n \leftarrow 0$

$\Theta_{root}, \Gamma_{root} \leftarrow determineRootJustifications($
  $\Theta^0_{U \sqsubseteq \bot}, \Gamma^0_{U \sqsubseteq \bot}, \ldots, \Theta^K_{U \sqsubseteq \bot}, \Gamma^K_{U \sqsubseteq \bot}))$

**while** not all $\Theta^k_{U \sqsubseteq \bot} \in \Theta_{root}, \Gamma^k_{U \sqsubseteq \bot} \in \Gamma_{root}$ are empty **do**
  $\mathcal{U}_n \leftarrow \emptyset$

  // Determine candidate $\Gamma$-sets and solve conflicts
  $\Gamma \leftarrow determineGammaCandidates(\Theta^k_{U \sqsubseteq \bot}, \Gamma^k_{U \sqsubseteq \bot})$
  **for all** $\Gamma^k_{U \sqsubseteq \bot} \in \Gamma$ **do**
    $\mathcal{U}_n \leftarrow chooseGammaAxiom(\mathcal{U}_0, \ldots, \mathcal{U}_{n-1}, \mathcal{U}_n, \Gamma^k_{U \sqsubseteq \bot})$
  **end for**

  // Remove chosen $\Gamma$-axioms from splitting sets.
  **for all** $\Gamma^k_{U \sqsubseteq \bot}, \Theta^k_{U \sqsubseteq \bot}$ **do**
    $\Gamma^k_{U \sqsubseteq \bot} \leftarrow \Gamma^k_{U \sqsubseteq \bot} \setminus \mathcal{U}_n \qquad \Theta^k_{U \sqsubseteq \bot} \leftarrow \Theta^k_{U \sqsubseteq \bot} \setminus \mathcal{U}_n$
  **end for**

  $resolveDerivedUnsatJustifications($
    $\Gamma, \Theta^0_{U \sqsubseteq \bot}, \Gamma^0_{U \sqsubseteq \bot}, \ldots, \Theta^K_{U \sqsubseteq \bot}, \Gamma^K_{U \sqsubseteq \bot})$

    // Resolve $\Theta$-axioms that are not in any $\Gamma$-set.
    for all $\Theta^k_{U \sqsubseteq \bot} : \Theta^k_{U \sqsubseteq \bot} \cap \Gamma^k_{U \sqsubseteq \bot} = \emptyset \land \Theta^k_{U \sqsubseteq \bot} \neq \emptyset$ do
        $\Theta_n \leftarrow chooseThetaAxiom(\mathcal{U}_0, \ldots, \mathcal{U}_n, \Theta^k_{U \sqsubseteq \bot})$
    end for

    // Resolve derived unsat justifications.
    $resolveDerivedUnsatJustifications($
        $\Theta, \Theta^0_{U \sqsubseteq \bot}, \Gamma^0_{U \sqsubseteq \bot}, \ldots, \Theta^K_{U \sqsubseteq \bot}, \Gamma^K_{U \sqsubseteq \bot})$

    // Remove chosen $\Theta$-axioms from splitting sets.
    for all $\Gamma^k_{U \sqsubseteq \bot}, \Theta^k_{U \sqsubseteq \bot}$ do
        $\Gamma^k_{U \sqsubseteq \bot} \leftarrow \Gamma^k_{U \sqsubseteq \bot} \setminus \mathcal{U}_n$
        $\Theta^k_{U \sqsubseteq \bot} \leftarrow \Theta^k_{U \sqsubseteq \bot} \setminus \mathcal{U}_n$
    end for
    $n \leftarrow updateIndex(n)$
end while

// move axioms with complex unsatisfiable concepts to partitions
$moveComplexUnsatConceptsToPartitions(\mathcal{T}, \mathcal{U}_0, \ldots, \mathcal{U}_N)$

// $\mathcal{T}_\triangle$ is the set of all axioms from $\mathcal{T}$ that are
// not contained a partition
$\mathcal{T}_\triangle = \mathcal{T} \setminus \bigcup_{n=0}^{N} \mathcal{U}_N$
```

contained in any Γ-set. As a result, we end up with a consistent Multi-View TBox $\mathcal{DT}$.

The subsequent sections of Section 4.4 describe in more detail how the single steps work and why they produce sound results. Several sub-algorithms for the function that Algorithm 1 calls are given along with an analysis of the complexity of the main Algorithm 1.

4.4.1 Resolving JUCs

Why resolving JUCs works.

The following observation seems trivial but is nonetheless important, since it describes the working principle why we can apply Default Logics for solving conflicts. Whenever two axioms for an arbitrary justification $J^{k_0}_{U_0 \sqsubseteq \bot}$ for $\mathcal{T} \models U_0 \sqsubseteq \bot$ are contained in two different partitions $\mathcal{U}_{n_0}$ and $\mathcal{U}_{n_1}$, $n_0 \neq n_1$, then $J^{k_0}_{U_0 \sqsubseteq \bot}$ is not a justification for $U_0 \sqsubseteq \bot$ neither w.r.t. $\mathcal{T} \setminus \mathcal{U}_{n_0}$ nor w.r.t. $\mathcal{T} \setminus \mathcal{U}_{n_1}$. The minimality constraint of justifications and monotonicity, even implies $J^{k_0}_{U_0 \sqsubseteq \bot}$ is not a justification for $U_0 \sqsubseteq \bot$ w.r.t. $\mathcal{T} \setminus \bigcup_{m=0}^{n} \mathcal{U}_m$.

Lemma 4. *Let $\mathcal{T}$ be a TBox and $\mathcal{U}_0, \ldots, \mathcal{U}_N$ be the partitions that result from Algorithm 1. If an axiom from an justification for an unsatisfiable concept $J^{k_0}_{U_0 \sqsubseteq \bot}$ for $\mathcal{T} \models U_0 \sqsubseteq \bot$ is contained in a partition $\mathcal{U}_{n_0}$, then $J^{k_0}_{U_0 \sqsubseteq \bot}$ is not a justification for $U_0 \sqsubseteq \bot$ w.r.t. $\mathcal{T} \setminus \bigcup_{m=0}^{n} \mathcal{U}_m$.*

However Lemma 4 provides the principles why the partitioning invalidates single justifications but we cannot conclude that $\mathcal{T} \setminus \bigcup_{m=0}^{n} \mathcal{U}_m \not\models U_0 \sqsubseteq \bot$. This conclusion we can draw only, if *all* justifications for $U_0 \sqsubseteq \bot$ are invalid w.r.t. $\mathcal{T} \setminus \bigcup_{m=0}^{n} \mathcal{U}_m$.

4.4.2 Candidates for $\mathcal{T}_\Delta$

Axioms that are valid for all partitions.

In case a justification $J^{k_0}_{U_0 \sqsubseteq \bot}$ has more than two axioms, we can even go one step further. Actually, it is not necessary that all axioms have to occur in different partitions. Indeed, it suffices that only two of them occur in two different partitions $\mathcal{U}_{n_0}$ and $\mathcal{U}_{n_1}$ for $n_0 \neq n_1$. On the other hand, when we add the axioms that we did not choose for $\mathcal{U}_{n_0}$ and $\mathcal{U}_{n_1}$ to *both* of these partitions, the $J^{k_0}_{U_0 \sqsubseteq \bot}$ is still invalid w.r.t. $\mathcal{T} \setminus \mathcal{U}_{n_0}$ and w.r.t. $\mathcal{T} \setminus \mathcal{U}_{n_1}$.

Lemma 5. *Let $\mathcal{T}$ be a TBox and $J^{k_0}_{U_0 \sqsubseteq \bot}$ be a root JUC for $\mathcal{T} \models U_0 \sqsubseteq \bot$. If one axiom $B \sqsubseteq A$ from a justification $J^{k_0}_{U_0 \sqsubseteq \bot}$ is contained in a partition $\mathcal{U}_{n_0}$, and one axiom $C \sqsubseteq D$ is contained in partition $\mathcal{U}_{n_1}$ for $n_0 \neq n_1$, then for all axioms $F \sqsubseteq E \in J^{k_0}_{U_0 \sqsubseteq \bot} \setminus \{B \sqsubseteq A, C \sqsubseteq D\}$ it holds that $J^{k_0}_{U_0 \sqsubseteq \bot}$ is not valid w.r.t.*
$\left(J^{k_0}_{U_0 \sqsubseteq \bot} \setminus \{B \sqsubseteq A, C \sqsubseteq D\}\right) \cup \left(\mathcal{T} \setminus \bigcup_{m=0}^{\max\{n_0,n_1\}} \mathcal{U}_m\right)$.

Lemma 5 describes the necessary condition for being able to add "surplus" axioms from an JUC to the Universal TBox $\mathcal{T}_\Delta$ instead of adding them to a partition $\mathcal{U}_n$ However, it remains to prove that $\mathcal{T} \setminus \bigcup_{m=0}^{N} \mathcal{U}_m \models E \sqcap F$ for all axioms $F \sqsubseteq E \in \mathcal{T}_\Delta$. We will give proof on that along with the proof of Theorem 6 in Appendix B.3.

4.4.3 Resolving Derived JUCs

The following observation explains why we have to consider all justifications, the root as well as the derived ones, when we use the sets from the unsat splitting for checking the satisfiability of $A \sqcap B$ w.r.t. $\mathcal{T} \setminus \bigcup_{m=0}^{n} \mathcal{U}_m$ to decide whether $B \sqsubseteq A$ is a candidate for the next partition $\mathcal{U}_{n+1}$.

Why resolving the root JUCs also resolves the dervied JUCs.

Lemma 6. *Let the partitions $\mathcal{U}_0, \ldots, \mathcal{U}_n$ be already computed according to Algorithm 1. The axioms of $\Theta^{k_0}_{U_0 \sqsubseteq \bot}$ can still contain concepts that are unsatisfiable w.r.t. $\mathcal{T} \setminus \bigcup_{m=0}^{n} \mathcal{U}_m$.*

Since Lemma 6 claims the potential existence of concepts that are still unsatisfiable, we proof Lemma 6 by giving an example.

Example 11 (Resolving derived JUCs)**.**
Consider the following TBox:

$$\mathcal{T} = \{B \sqsubseteq A, C \sqsubseteq B, C \sqsubseteq \neg A, D \sqsubseteq C, E \sqsubseteq D, F \sqsubseteq E, F \sqsubseteq \neg A\}$$

It has the root JUC:

$$\begin{aligned} J^0_{C \sqsubseteq \bot} &= \overbrace{\{B \sqsubseteq A\}}^{\Theta} \cup \overbrace{\{C \sqsubseteq B, C \sqsubseteq \neg A\}}^{\Gamma} \\ J^1_{F \sqsubseteq \bot} &= \{B \sqsubseteq A, C \sqsubseteq B, E \sqsubseteq D\} \cup \{F \sqsubseteq E, F \sqsubseteq \neg A\} \end{aligned}$$

Considering only the root JUC when choosing the axioms for the first partition $\mathcal{U}_0$ we would, according to Algorithm 1, choose all those axioms that do occur in some Θ-set but not in any Γ-set.[4] *Hence, we would choose axiom $E \sqsubseteq D$ for the first partition $\mathcal{U}_0$. Yet, $\mathcal{T} \models E \sqsubseteq \bot$ with the justification*

$$J^2_{F \sqsubseteq \bot} = \{B \sqsubseteq A, C \sqsubseteq B\} \cup \{E \sqsubseteq D\}$$

Hence we must not add $E \sqsubseteq D$ to $\mathcal{U}_0$, because the condition for Default Logics is not fulfilled: $\mathcal{T} \not\models A \sqcap B$. In fact, the axiom $E \sqsubseteq D$ is part of a Γ-set for a derived justification. If we had checked also the Γ-sets of the derived justifications, we would have prevented the incorrect addition of the axiom $E \sqsubseteq D$ to $\mathcal{U}_0$.

Including the derived JUCs when choosing axioms is important.

Example 11 makes clear why we have to include the derived JUC when choosing axioms for a partition $\mathcal{U}_{n+1}$. As long as the corresponding justification is valid w.r.t. a TBox $\mathcal{T}'$, we can find concepts that are unsatisfiable w.r.t. $\mathcal{T}'$ by checking the Γ-sets of *all* JUC for $\mathcal{T}'$. If we identify $\mathcal{T}'$ by $\mathcal{T} \setminus \bigcup_{m=0}^{n} \mathcal{U}_m$ we can use the Γ-sets for checking whether one of the concepts in the signature of an axiom $B \sqsubseteq A \in \mathcal{T} \setminus \bigcup_{m=0}^{n} \mathcal{U}_m$ is unsatisfiable w.r.t. $\mathcal{T} \setminus \bigcup_{m=0}^{n} \mathcal{U}_m$. Hence, the Θ-sets provide candidate axioms $B \sqsubseteq A$ for $\mathcal{U}_{n+1}$ whereas the Γ-sets, in turn, give proof whether the candidates fulfill the condition of Default Logics, i.e. $\mathcal{T} \setminus \bigcup_{m=0}^{n} \mathcal{U}_m \models A \sqcap B$.

[4] Note that Γ is always empty in the first iteration of Algorithm 1

Algorithm 2 gives a proposal for how to implement resolving derived justifications. Resolving a JUC is done by removing all axioms from both of its splitting sets. This way, they cannot contribute axioms to the partitions $\mathcal{U}_0, \ldots, \mathcal{U}_N$.

From the set of currently valid (i.e. not yet resolved) JUCs all those JUCs are removed that share at least one axiom with some already resolved Γ-sets. Both, the currently valid JUCs as well as the already resolved Γ-sets are given as an input to the algorithm in the form of the corresponding splitting sets. If one of the currently valid JUCs shares an axiom with one of the already resolved Γ-sets, it will be resolved as well.

Algorithm 2 Resolve those derived JUCs that depend on a given set of JUCs.

Function: $resolveDerivedUnsatJustifications($
$\Gamma, \Theta^0_{U\sqsubseteq\bot}, \Gamma^0_{U\sqsubseteq\bot} \ldots, \Theta^K_{U\sqsubseteq\bot}, \Gamma^K_{U\sqsubseteq\bot})$

Input:
$\Gamma = \{\Gamma^{k_j}_{U\sqsubseteq\bot}\}$ candidate splitting sets that have been resolved
$\Theta^k_{U\sqsubseteq\bot}, \Gamma^k_{U\sqsubseteq\bot}, k = 0, \ldots, K$ splitting sets to be resolved

```
// invalidate all derived Θ- and Γ-sets
// that share an axiom with a resolved root Γ-set
for all $\Theta^k_{U\sqsubseteq\bot}, \Gamma^k_{U\sqsubseteq\bot}$ do
    for all $\Gamma^{k_j}_{V\sqsubseteq\bot}$ do
        if $J^k_{U\sqsubseteq\bot}$ is derived and $\Theta^k_{U\sqsubseteq\bot} \cap \bigcup_j \Gamma^{k_j}_{V\sqsubseteq\bot} \neq \emptyset$ then
            $\Theta^k_{U\sqsubseteq\bot}, \Gamma^k_{U\sqsubseteq\bot} \leftarrow \emptyset$
        end if
    end for
end for
```

4.4.4 Determine Γ-sets for Resolving

Although we resolved a root JUC for some unsatisfiable concept U_0 in step n, we saw in Lemma 6 there may still exist root JUC which state that U_0 is unsatisfiable w.r.t. $\mathcal{T} \setminus \bigcup_{m=0}^{n} \mathcal{U}_m$. When resolving $\Gamma^{k_0}_{U_0 \sqsubseteq \bot}$ of a root JUC, we must indeed resolve *all* $\Gamma^{k_j}_{U_0 \sqsubseteq \bot}$ for the unsatisfiable concept U_0. If we do not do so, we may have in the partition $\mathcal{U}_{n+1}$ an axiom $D \sqsubseteq C$ for which C is not satisfiable. This, in turn, contradicts the condition given by Default Logics that $(\mathcal{T}_\Delta \cup \mathcal{U}_{n+1}) \not\models C \sqcap D \sqsubseteq \bot$. We illustrate this by the following example:

Example 12. *Assume the following TBox:*

$$\begin{aligned}\mathcal{T} \quad = \quad & \{B \sqsubseteq A, C \sqsubseteq B, C \sqsubseteq \neg A, D \sqsubseteq C, E \sqsubseteq D,\\ & E \sqsubseteq \neg C, F \sqsubseteq E, G \sqsubseteq F, G \sqsubseteq \neg E, I \sqsubseteq G,\\ & H \sqsubseteq A, I \sqsubseteq H, I \sqsubseteq \neg A\}\end{aligned}$$

The first two partitions and the remainder set applying Algorithm 1 and choosing all axioms are

$$\begin{aligned}\mathcal{T}_\Delta &= \{I \sqsubseteq G\}\\ \mathcal{U}_0 &= \{B \sqsubseteq A, H \sqsubseteq A\}\\ \mathcal{U}_1 &= \{C \sqsubseteq B, C \sqsubseteq \neg A, D \sqsubseteq C, I \sqsubseteq H, I \sqsubseteq \neg A\}\\ \mathcal{T} \setminus (\mathcal{U}_0 \cup \mathcal{U}_1) &= \{E \sqsubseteq D, E \sqsubseteq \neg C, F \sqsubseteq E, G \sqsubseteq F,\\ & \quad G \sqsubseteq \neg E, I \sqsubseteq G\}\end{aligned}$$

Yet, we have that $J^0_{I \sqsubseteq \bot} = \{F \sqsubseteq E, G \sqsubseteq F, G \sqsubseteq \neg E, I \sqsubseteq G\}$ *and that* $J^0_{I \sqsubseteq \bot} \subset \mathcal{T} \setminus (\mathcal{U}_0 \cup \mathcal{U}_1)$.[5] *This means that we have evidence for* $\mathcal{T} \setminus (\mathcal{U}_0 \cup \mathcal{U}_1) \models I \sqsubseteq \bot$. *Consequently,* $\mathcal{T} \setminus (\mathcal{U}_0 \cup \mathcal{U}_1) \not\models I \sqcap H$. *Because of monotonicity,* $\mathcal{T} \setminus \mathcal{U}_0 \not\models I \sqcap H$, *too. Hence, the partition* $\mathcal{U}_1$ *contains an axiom for which the condition given by Default Logics is not fulfilled:* $I \sqsubseteq H \in \mathcal{U}_1$, *but* $\mathcal{T} \setminus \mathcal{U}_0 \not\models I \sqcap H$.

[5] We can rewrite $\mathcal{T} \setminus (\mathcal{U}_0 \cup \mathcal{U}_1)$ as $\mathcal{T}_\Delta \cup \bigcup_{m=2}^{N} \mathcal{U}_m$ We may not know the actual value of N yet, but we know the contents of $\bigcup_{m=2}^{N} \mathcal{U}_m$

We may resolve in step $n+1$ the Γ-sets that belong to root JUC for a concept U_0 only if *all* justifications, i.e. also the derived ones, for U_0 have been solved in step n. This is the case when all root unsat $\Theta^k_{U_0 \sqsubseteq \bot}$ are empty or one axiom from a derived unsat $\Theta^k_{U_0 \sqsubseteq \bot}$ is contained in one of the already computed partitions $\mathcal{U}_m$ for $m \leq n$.

As a direct consequence from Lemma 6, we either resolve *all* root unsat Γ-sets for U_0 in step $n+1$ or none of them. This is the case when *all* Θ-sets that belong to that unsatisfiable concept U_0 have been resolved—this includes both Θ-sets from the root as well as from the derived JUC for $\mathcal{T} \models U_0 \sqsubseteq \bot$.

Lemma 7. *Let U_0 be an unsatisfiable concept w.r.t. $\mathcal{T}$ that has the $j = 0, \ldots, J$ root JUC $J^{k_j}_{U_0 \sqsubseteq \bot}$ w.r.t. $\mathcal{T}$. In the $n+1$-th step of applying Algorithm 1 to $\mathcal{T}$ either $\bigcup_{j=0}^{J} \Gamma^{k_j}_{U_0} \subseteq (\mathcal{T}_\Delta \cup \mathcal{U}_{n+1})$ or $\bigcup_{j=0}^{J} \Gamma^{k_j}_{U_0} \cap (\mathcal{T}_\Delta \cup \mathcal{U}_{n+1}) = \emptyset$.*

Algorithm 3 provides a suggestion how Lemma 7 can be implemented.

4.4.5 Resolving Axioms from Θ-sets

Having resolved all possible Γ-sets we resolved the derived justifications that depended on these Γ-sets, as well. By removing axioms from Γ-sets of the derived justifications, it can happen that axioms are still contained in a Θ-set of a root JUC but no more contained in any Γ-set. If an axiom $B \sqsubseteq A$ is not contained in any Γ-set w.r.t. $\mathcal{T} \setminus \bigcup_{m=0}^{n} \mathcal{U}_m$ we have no more evidence that B is entailed unsatisfiable w.r.t. $\mathcal{T} \setminus \bigcup_{m=0}^{n} \mathcal{U}_n$. We can hence conclude that $\mathcal{T} \setminus \bigcup_{m=0}^{n} \mathcal{U}_n \models A \sqsubseteq B$.

Lemma 8. *Let $\mathcal{U}_0, \ldots, \mathcal{U}_{n+1}$ and $\mathcal{T}_\Delta$ be the partitions and the Universal TBox that result from Algorithm 1 after Line 23. If for an axiom $B \sqsubseteq A \in \Theta^{k_0}_{U_0 \sqsubseteq \bot}$ it holds that $B \sqsubseteq A \notin \bigcup_{k=0}^{K} \Gamma^k_{V \sqsubseteq \bot}$, then $\mathcal{T} \setminus \bigcup_{m=0}^{n} \mathcal{U}_n \models B \sqsubseteq A$.*

As a direct consequence from Lemma 8 $B \sqsubseteq A$ is a candidate for $\mathcal{U}_{n+1}$. Furthermore, if for some $\Theta^{k_0}_{U_0 \sqsubseteq \bot}$ *more than one axiom* is no more contained neither in any Γ-set nor the dependent splitting sets, then the axioms from $\Theta^{k_0}_{U_0 \sqsubseteq \bot}$ are also candidates for $\mathcal{T}_{\triangle}$.

Algorithm 3 Determine the candidates to be resolved in the current iteration step from Γ_{root}, i.e. from the Γ-sets of the root JUCs.

Function: $\Gamma \leftarrow$ $\mathit{determineGammaCandidates}(\Theta^0_{U \sqsubseteq \bot}, \Gamma^0_{U \sqsubseteq \bot} \ldots, \Theta^K_{U \sqsubseteq \bot}, \Gamma^K_{U \sqsubseteq \bot})$

Input:

Splitting sets $\Theta^k_{U \sqsubseteq \bot}, \Gamma^k_{U \sqsubseteq \bot}$ for all $k = 0, \ldots, K$ JUCs for a TBox $\mathcal{T}$.

Output:

Candidate Γ-sets for solving

```
// collect all possible candidate sets
// grouped by the unsatisfiable concept
for all $U$ do
    $\Gamma_U \leftarrow \{\Gamma^k_{U \sqsubseteq \bot} | \Gamma^k_{U \sqsubseteq \bot} \neq \emptyset\}$
end for
// remove groups of candidates
// for which one member's $\Theta$-set is not empty
for all $\Gamma^k_{U \sqsubseteq \bot}$ do
    if $\Theta^k_{U \sqsubseteq \bot} \neq \emptyset$ then
        $\Gamma_U \leftarrow \emptyset$
    end if
end for

// collect all candidates that belong to a
// root JUC as a result
$\Gamma \leftarrow \{\Gamma^k_{U \sqsubseteq \bot} \in \bigcup_U \Gamma_U | J^k_{U \sqsubseteq \bot}$ is a root JUC$\}$
```

4.4.6 Solving Axioms With Complex Unsatisfiable Concepts

After finishing computing the partitions, it is possible that the set $\mathcal{T} \setminus \bigcup_{n=0}^{N} \mathcal{U}_n$ contains axioms whose extended signature contains complex concepts that are unsatisfiable w.r.t. $(\mathcal{T} \setminus \bigcup_{n=0}^{N} \mathcal{U}_n) \cup \mathcal{U}_n$ for some partition n.[6] We must move each of these axioms to a particular partition before we can compute the Universal TBox as $\mathcal{T}_\Delta = \mathcal{T} \setminus \bigcup_{n=0}^{N} \mathcal{U}_n$. The procedure is described by Algorithm 4. We check each partition $\mathcal{U}_n$ for $n = N, \ldots, 0$ whether it contains the declaration for a complex concept U. Since all of these declarations $U \sqsubseteq \top$ must occur in Γ-sets, we know that $\mathcal{T} \models U \sqsubseteq \bot$. On the other hand, it is possible that $U \sqsubseteq \top$ is contained in some Θ-sets. This is, for example, the case when there exists another complex unsatisfiable concept $\tilde{U}$ with $\tilde{U} \neq U$ and $U \sqsubseteq \tilde{U} \in \mathcal{T}$. We have hence to start from the last partition. If axioms with U in the extended signature are added to partition $\mathcal{U}_n$ we hence ensure that all such concepts $\tilde{U}$ are put in a partition $\mathcal{U}_m$ with $m \leq n$. This, in turn, ensures that neither $(\mathcal{T}_\Delta \cup \mathcal{U}_n) \models U \sqsubseteq \bot$ nor $(\mathcal{T}_\Delta \cup \mathcal{U}_n) \models \tilde{U} \sqsubseteq \bot$.

Example 13 (Move complex unsatisfiable concepts)**.**
Assume the TBox:

$$\mathcal{T} = \{B \sqsubseteq A, C \sqsubseteq \neg A, E \sqsubseteq B \sqcap C, B \sqcap C \sqcap F \sqsubseteq D\}$$

Applying Algorithm 1 until Line 43 results into the partitions

$$\begin{aligned} \mathcal{U}_0 &= \{B \sqsubseteq A, C \sqsubseteq \neg A\} \\ \mathcal{U}_1 &= \{B \sqcap C \sqsubseteq \top, B \sqcap C \sqcap F \sqsubseteq \top\} \\ \mathcal{T} \setminus (\mathcal{U}_0 \cup \mathcal{U}_1) &= \{E \sqsubseteq B \sqcap C, B \sqcap C \sqcap F \sqsubseteq D\}) \end{aligned}$$

[6] Note that $\mathcal{T} \setminus \bigcup_{n=0}^{N} \mathcal{U}_n$ is the candidate set for $\mathcal{T}_\Delta$.

Proceeding Algorithm 1 further we obtain

$$\begin{aligned} \mathcal{U}_0 &= \{B \sqsubseteq A, C \sqsubseteq \neg A\} \\ \mathcal{U}_1 &= \{B \sqcap C \sqsubseteq \top, B \sqcap C \sqcap F \sqsubseteq \top, \\ &\quad E \sqsubseteq B \sqcap C, B \sqcap C \sqcap F \sqsubseteq D\} \\ \mathcal{T} \setminus (\mathcal{U}_0 \cup \mathcal{U}_1) &= \{\} \end{aligned}$$

If we had not added the axioms that contain complex unsatisfiable concepts in their extended signature, in Example 13 we would end up with $\mathcal{T}_\Delta = \{E \sqsubseteq B \sqcap C, B \sqcap C \sqcap F \sqsubseteq D\}$. Yet, $(\mathcal{T}_\Delta \cup \mathcal{U}_0) \models B \sqcap C \sqcap F \sqsubseteq \bot$. Even worse we would have that $(\mathcal{T}_\Delta \cup \mathcal{U}_0) \models E \sqsubseteq \bot$. We can avoid this by executing Algorithm 4.

4.4.7 Consistency

The most important property of the Defaults-Splitting-Δ-Operator is that the resulting Default-TBox is consistent. We therefore formalize the necessary conditions. To ensure that always two axioms of a root JUC are contained in two different partitions, we require that in every step of Algorithm 1 we can choose at least one axiom from a Γ-set of a root JUC. Furthermore, we require that, in the first step, we can choose at least one Θ-axiom that is not contained in any Γ-set. Finally, the input TBox must be acyclic but must not contain neither logical nor direct contradictions.

Theorem 6. *The Default TBox produced by Algorithm 1 is consistent, if the following holds:*

$$\begin{aligned} \left| chooseThetaAxiom(\mathcal{U}_n, \mathcal{T}_\Delta, \Theta^k_{U \sqsubseteq \bot})\right| &\geq 1 \quad if \quad n = 0. \\ \left| chooseGammaAxiom(\mathcal{U}_n, \mathcal{T}_\Delta, \Gamma^k_{U \sqsubseteq \bot})\right| &\geq 1 \quad if \quad n > 0. \\ updateIndex(n) &= n + 1 \end{aligned}$$

The index update function simply states that in the next step we start with a new empty partition. We will see in Section 5.2 that we can indeed have only two partitions and the resulting Multi-View-TBox $\mathcal{DT}$ still be consistent but we then give up some properties on $\mathcal{DT}$.

The proof for Theorem 6 can be found in Appendix B.3. Here, we will illustrate how the Aspect TBoxes $\mathcal{U}_0, \ldots, \mathcal{U}_N$ are constructed. The main idea behind Algorithm 1 is to successively add at least one axiom of each $\Theta_{U \sqsubseteq \bot}$ to $\mathcal{U}_n$ and at least one axiom of each $\Gamma_{U \sqsubseteq \bot}$ to $\mathcal{U}_{n+1}$. The remaining axioms are then added to $\mathcal{T}_\Delta$.

4.4.8 Uniqueness of Solutions

The choice of axioms determines which axioms are treated as Defaults and which are treated as crisp DL axioms. It was shown in Lukasiewicz [2008] that for a Universal TBox $\mathcal{T}_\Delta$ and a set of Defaults (called conditional constraints there), the resulting partitions are unique.

Corollary 1. *Each choice made in Algorithm 1 produces a unique Multi-View-TBox.*

As a consequence, Algorithm 1 produces unique solutions, i.e. it is deterministic. Yet, the oracle for choosing which axioms are considered as Defaults and which are considered crisp may be non-deterministic. Although there can exist many alternatives the oracle can choose from, Corollary 1 states that for any choice the oracle actually makes, there exists only one possible solution how to partition the set of Defaults. For any choice of the oracle there exists exactly one possible Multi-View-TBox.

4.4.9 Complexity

The complexity of finding a Multi-View TBox for a TBox $\mathcal{T}$ is dominated by finding all JUC which depends on the underly-

Algorithm 4 Remove those axioms that contain an unsatisfiable complex concept in their extended signature.

Function:
$moveComplexUnsatConceptsToPartitions(\mathcal{T}, \mathcal{U}_0, \ldots, \mathcal{U}_N)$

Input:
TBox $\mathcal{T}$
Partitions $\mathcal{U}_0, \ldots, \mathcal{U}_N$

```
for $n = N, \ldots, 0$ do
    for $\tau \in \mathcal{U}_n$ do
        // check whether we solved a complex unsatisfiable
        // concept $U$ in the current partition $\mathcal{U}_n$
        if $\tau = U \sqsubseteq \top$ and $U$ is complex then
            // add all axioms $\rho$ that contain the resolved complex concept
            // $U$ in the extended signature to the current partition $\mathcal{U}_n$
            for $\rho \in \mathcal{T}$ do
                if $U \in \widetilde{sig}(\rho)$ then
                    $\mathcal{U}_n \leftarrow \mathcal{U}_n \cup \{\rho\}$
                end if
            end for
        end if
    end for
end for
```

ing expressive power of the ontology. As we saw in Section 2.1, for expressive DLs like $\mathcal{SROIQ}$ this problem is 2NEXPTIME-complete in the number of axioms in the TBox.

Set operations for finding partitions

Finding partitions does not require any reasoning but only set operations. For constructing these sets, we computed the Unsat-Splitting. We have to perform the check whether the left-hand-side of an axiom is an unsatisfiable concept for all axioms of each

JUC, not only the root ones. Let us denote the number of axioms in the JUCs by L, and the number of axioms in the root JUCs only by $\tilde{L}$. In the worst case, $L = |\mathcal{T}|$, i.e. all axioms of the TBox are involved in some JUC. Let us further denote the number of JUCs by K and the number of root JUCs by $\tilde{K}$.

We have to determine the dependency between the different justifications. This can be accomplished by a dependency graph whose root nodes are the root JUCs. Building this graph requires $K \times L$ set comparison operations in the worst case.

For computing the first partition we have to compare each Θ-set with each Γ-set of the root JUCs. This requires $\tilde{K}^2$ set comparison operations. In the worst case, only one root JUC contributes an axiom to $\mathcal{U}_0$ leaving more axioms for comparison when computing further partitions. As a result, we have to perform $\tilde{K} \times \tilde{K}$ set comparisons for determining $\mathcal{U}_0$. Hence, computing $\mathcal{U}_1$ requires a maximum number of $(\tilde{K} - 1)^2$ set comparison operations.

We can end up with a maximum number of $\tilde{K}$ partitions $\mathcal{U}_n$ for $n = 0, \ldots, \tilde{K}$. This requires $\tilde{K}^2 + (\tilde{K} - 1)^2 + \ldots = \tilde{K}(\tilde{K} + 1)(2\tilde{K} + 1)/6$ computations, which is bounded by $\mathcal{O}(\tilde{K}^3)$. Note that in the best case, we only have two partitions such that the number of computations is bounded by $\mathcal{O}(\tilde{K}^2)$.

The sets can be coded as hash-sets which allows for constant time for accessing elements. As a result, the number of access operations for performing a comparison of two sets S_1 and S_2 is bounded by $\mathcal{O}(\max |S_1|, |S_2|)$. In the worst case, the number of axioms in the root JUCs correlates with the number of axioms in the ontology, which increases the computational worst-case complexity from $\mathcal{O}(\tilde{K}^3)$ to $\mathcal{O}(\tilde{K}^4)$.

We should note that in the ontologies that were used for the experiments the number of axioms in the root JUCs was very small compared to the number of axioms in the ontology. The same held for the number of justifications and the number of par-

Finding JUCs dominates complexity.

titions. Finding a Multi-View-TBox by Algorithm 1 is dominated by the procedure of finding JUCs.

We should also stress out that Algorithm 1 is deterministic whenever the choice function which of the axioms go into the single partitions and which that go into $\mathcal{T}_\Delta$ is deterministic. This is the case for the All-Defaults-Δ-Operator whereas the choice for the Minimal-Defaults-Δ-Operator is irreversible but not deterministic. Yet, all presented variants of the Δ-Operator produce a consistent Multi-View TBox.

5

The Unsat-Splitting-Δ-Operators

Using the Unsat-Splitting defined in Section 4.3 we define a number of Splitting-Δ-Operators, all of which result in a consistent Multi-View TBox. However, each of these operators has different properties. Constraining the computation of Aspect TBoxes using Default Logics, we have a guided procedure with unique solution but risk to invalidate too many entailments. This approach can be refined in a way that fewer axioms are put into the Aspect TBoxes. On the one hand, this results in fewer entailments invalidated, on the other hand, the solution is not unique anymore.

When $\Delta(\mathcal{T})$ does not need to provide Default Logics reasoning, we can show that there exists a Splitting Δ-operator that produces a consistent Multi-View TBox with only exactly two Aspect TBoxes. As the previous approach it is non-deterministic but is expected to invalidate even fewer entailments.

We use the Example 10 to illustrate how the Multi-View TBox is constructed for each approach. The construction, in turn, is described by Algorithm 1 and differs for each approach only slightly.

The differences are described in the Sections corresponding the respective approach.

Plan of This Chapter

We start with the Δ-Operator that transforms all axioms of all root JUCs in Section 5.1.1 and continue with its improved version w.r.t. the number of axioms in Section 5.1.2. In the subsequent Section 5.2 we present the optimization regarding the number of partitions and sketch a possible iterative solution in Section 5.3. We close this chapter with an overview over alternative approaches for a Splitting-Δ-Operator in Section 5.4 and how the Splitting-Δ-Operator can be defined on a lattice in Section 5.5.

Finding optimal solutions for the non-deterministic approaches is subject to Chapter 6

5.1 The Defaults-Splitting-Δ-Operator

As we saw in Section 2.5 Default Logics provide a mechanism for solving conflicts on a set of DL axioms *without having to remove any axiom*. These axioms are treated as Defaults, which, in turn, provides a partitioning algorithm that, for a TBox with unsatisfiable concepts, produces a unique Multi-View-TBox $\mathcal{DT}$ as a result. We can show that when we add at least two axioms of all root JUCs to two different Aspect TBoxes $\mathcal{U}_n$ the resulting Multi-View TBox $\mathcal{DT}$ is consistent. Furthermore, we can use the Unsat Splitting from Section 4.3 to compute these partitions without the need for additional satisfiability checks.

5.1.1 All-Defaults-Splitting-Δ-Operator

The simple-most strategy for the oracle is to choose all candidates as defaults. Since following this strategy, all axioms of all root JUC are considered as Defaults, we refer to the induced operator as the *All-Defaults-Splitting-Δ-Operator*.

The oracle is defined as follows:

$$\begin{aligned}
chooseThetaAxiom(\mathcal{U}_0,\ldots,\mathcal{U}_n,\mathcal{T}_\Delta,\Theta^k_{U \sqsubseteq \bot}) &= \Theta^k_{U \sqsubseteq \bot}\\
chooseGammaAxiom(\mathcal{U}_0,\ldots,\mathcal{U}_n,\mathcal{T}_\Delta,\Gamma^k_{U \sqsubseteq \bot}) &= \Gamma^k_{U \sqsubseteq \bot}\\
updateIndex(n) &= n+1
\end{aligned}$$

For a TBox $\mathcal{T}$ there exists only one solution how the All-Defaults-Splitting-Δ-Operator transforms $\mathcal{T}$ into a Multi-View-TBox. The oracle chooses all axioms of all root JUC to be considered as Defaults. We know from Corollary 1 that for a given set of defaults there exists only one solution how to partition these according to Lehmann's Default Logics.

First Approach: All axioms are Defaults.

The unambiguity of the solution to $\Delta(\mathcal{T})$, however, comes at a price: We risk invalidating unnecessarily many entailments. We know from Theorem 6 that it suffices to transform only two axioms from each root JUC to obtain a consistent Multi-View-TBox. In that case, the solution to $\Delta(\mathcal{T})$ is not deterministic anymore and requires further optimization. We will discuss this approach in Section 5.1.2

5.1.2 Minimal-Defaults-Splitting-Δ-Operator

Putting all axioms of all root JUCs in the Aspect TBoxes results in a unique consistent Multi-View-TBox. Adding fewer axioms to the Aspect TBoxes $\mathcal{U}_n$ while leaving more axioms to $\mathcal{T}_\Delta$, the resulting Multi-View TBox $\mathcal{DT}$ is expected to invalidate fewer entailments while still being consistent (Theorem 6). This solu-

Second Approach: As few axioms as possible are Defaults.

tion, however, requires an oracle that makes a non-deterministic choice.

Furthermore, the resulting Default-TBox can contain ABox conflicts, i.e. an instance a can be asserted a concept $A(a)$ and its complement $\neg A(a)$. This does not cause problems for the reasoning process but leads to ambiguous results. This issue is investigated in more detail in Section 6.2

We define the oracle for the *Minimal-Defaults-Splitting-Δ-Operator* by a random choice:

$$\Big| \quad chooseThetaAxiom(\mathcal{U}_0, \ldots, \mathcal{U}_n, \mathcal{T}_\Delta, \Theta^k_{U \sqsubseteq \bot})\Big| \\ \qquad = rand(\Theta^k_{U \sqsubseteq \bot} \setminus \bigcup_{m=0}^{n} \mathcal{U}_m)$$

$$\Big| \quad chooseGammaAxiom(\mathcal{U}_0, \ldots, \mathcal{U}_n, \mathcal{T}_\Delta, \Theta^k_{U \sqsubseteq \bot})\Big| \\ \qquad = rand(\Gamma^k_{U \sqsubseteq \bot} \setminus \bigcup_{m=0}^{n} \mathcal{U}_m)$$

$$updateIndex(n) = n + 1$$

Note that the actual choice depends on a random process, but we know from Corollary 1 that each random choice determines a unique solution.

5.1.3 Comparison

Comparing the first and the second approach.

We will now explain how the All-Defaults-Splitting-Δ-Operator and the Minimal-Defaults-Splitting-Δ-Operator work according to Example 10. We will also compare the differences between the two when going through the single processing steps. The single possible solution for the first operator and its processing steps is illustrated in Table 5.1. We present two possible solutions for the latter operator in Table 5.2 and Table 5.3.

Initialization

The initialization step is equal for both approaches. We compute the dependencies between the JUCs and initialize the counter n to 0.

Iteration 1

In the first iteration, we have not yet resolved any conflicts. There are no Γ-candidates that could be resolved (Line 12). We hence skip some of the following steps and continue with resolving the Θ-candidates (Line 26). In our case, the only axioms that are contained solely in a Θ-set but not in any Γ-set are τ_1 and τ_0.

Here, the first difference takes place. The All-Defaults-Splitting-Δ-Operator puts both axioms into the first partition $\mathcal{U}_0$. The Minimal-Defaults-Splitting-Δ-Operator, in contrast, only chooses one of them. Let's assume the oracle chooses axiom τ_1 to be contained in $\mathcal{U}_0$ and τ_0 to be contained in $\mathcal{T}_\Delta$.

None of the justifications is completely empty, so we increment the counter n by one and proceed to the next iteration step.

Iteration 2

In the second iteration, we know that we removed an axiom from $\Theta^1_{C\sqsubseteq\bot}$. Now we can determine Γ-candidates (Line 12). Since we only solved the Θ-set for exactly one root JUC we will find only one Γ-candidate for solving, in particular $\Gamma^1_{C\sqsubseteq\bot}$.

The set $\Gamma^1_{C\sqsubseteq\bot}$ consists of two axioms, namely τ_2 and τ_3. Again, the All-Defaults-Splitting-Δ-Operator chooses both of them for the partition $\mathcal{U}_1$. In contrast to that, the oracle of the Minimal-Defaults-Splitting-Δ-Operator selects one of them for $\mathcal{U}_1$. It does, however, not put the other axiom directly into $\mathcal{T}_\Delta$. This can be seen from Table 5.2. Assume that we chose τ_3 for $\mathcal{U}_1$ and put τ_2 into $\mathcal{T}_\Delta$. In that case we risk to empty a potential Θ-set for the next iteration. All those axioms that we did not choose for any partition in the following are put into the Universal TBox $\mathcal{T}_\Delta$, anyway. Now that we processed $\Gamma^1_{C\sqsubseteq\bot}$, we can resolve the derived JUCs (Line 23).

We proceed with determining the Θ-candidates. We have to distinguish between the All-Defaults-Splitting-Δ-Operator and the Minimal-Defaults-Splitting-Δ-Operator. In the first case the only axiom which is contained in a Θ-set but not in a Γ-set is τ_4 whereas in the latter case we can also choose from τ_2. Because we resolved $\Gamma^1_{C \sqsubseteq \bot}$ in the first part of the current iteration, τ_2 is not contained in any Γ-set anymore.

Again, applying the first operator we choose τ_4 for $\mathcal{U}_1$. We assume that the oracle for the second operator chooses axiom τ_2. There are no more dependent justifications to resolve, we can proceed with the next iteration. Before we do so, let us make a remark on the choice which the oracle of the Minimal-Defaults-Splitting-Δ-Operator made.

The choice of the oracle was not minimal. The situation is illustrated in Table 5.3. When resolving $\Gamma^1_{C \sqsubseteq \bot}$, we could have chosen either τ_2 or τ_3. Since τ_2 is contained in $\Theta^3_{D \sqcap C \sqsubseteq \bot}$, we could have solved this splitting set along with resolving $\Gamma^1_{C \sqsubseteq \bot}$. Hence, the choice becomes minimal, if the oracle decides to choose τ_2 to be contained in $\mathcal{U}_1$ instead of choosing τ_3 first and still having to choose τ_2 later.

Iteration 3

In the third iteration we resolve $\Gamma^3_{D \sqcap C \sqsubseteq \bot}$ for which we in any case have only one option. We add axiom τ_7 to $\mathcal{U}_2$. Since now all splitting sets of the root JUCs are solved, we exit the while-loop.

Post-Processing

We now have to add those axioms which contain an unsatisfiable complex concept in their extended signature to those partitions that contain a declaration axiom (Line 43). This provides the same results for both operators. We find that τ_6 contains $E \sqcup C$ in the signature which is a complex concept and unsatisfiable w.r.t. $\mathcal{T}$. Yet, there is no partition that contains the axiom $E \sqcup C \sqsubseteq \top$. For the complex concept $D \sqcap C$ which is also unsatisfiable w.r.t. $\mathcal{T}$ we have that partition $\mathcal{U}_2$ contains $D \sqcap C \sqsubseteq \top$. Furthermore, we have that $D \sqcap C \in \widetilde{sig}(\tau_9)$ As a consequence, we add τ_9 to $\mathcal{U}_2$.

Finally, we set $\mathcal{T}_\Delta = \mathcal{T}_\Delta \cup (\mathcal{T} \setminus \bigcup_{n=0}^{2} \mathcal{U}_n)$ and add all those axioms to $\mathcal{T}_\Delta$ that were not contained in a root JUC or not chosen to be part of a partition. We obtain $\mathcal{T}_\Delta = \{\tau_0, \tau_5, \tau_8\}$ for the All-Defaults-Splitting-Δ-Operator whereas we obtain $\mathcal{T}_\Delta = \{\tau_0, \tau_4, \tau_5, \tau_8\}$ and $\mathcal{T}_\Delta = \{\tau_0, \tau_3, \tau_4, \tau_5, \tau_8\}$ for the two choices of the Minimal-Defaults-Splitting-Δ-Operator.

We can see that even non-optimal choices for the Minimal-Defaults-Δ-Operator potentially invalidate fewer entailments than the All-Defaults-Splitting-Δ-Operator.

5.2 Minimal-Partitions-Splitting-Δ-Operator

Third Approach: Create as few partitions as possible.

Default TBoxes fulfill the properties of Lehmann's Default Logics (cf Section 2.5), but fulfilling these properties is not required in every case. If we are just interested in valid entailments, we do not care about specificity of information. In that case we define two disjoint Aspect TBoxes $\mathcal{U}_0$ and $\mathcal{U}_1$, each of which contains one axiom from each root JUC.

We may hence be able to solve arbitrary, for example circular, conflicts in the TBox. Yet, there is no guarantee that we can find $\mathcal{U}_0$ and $\mathcal{U}_1$ such that the corresponding Multi-View TBox is consistent.

We can indeed show that it is possible to construct a consistent Multi-View TBox with exactly two Aspect TBoxes when using the All-Defaults-Splitting-Δ-Operator or the Minimal-Defaults-Splitting-Δ-Operator.

As in the case of the Minimal-Defaults-Splitting-Δ-Operator, the resulting Multi-View TBox can contain ABox conflicts, i.e. an instance a can be asserted a concept $A(a)$ and its complement $\neg A(a)$. This does not cause problems for the reasoning process but leads to ambiguous results. This issue is investigated in more detail in Section 6.2

n		$J^1_{C\sqsubseteq\bot}$	$J^2_{D\sqcap C\sqsubseteq\bot}$	$J^3_{D\sqcap C\sqsubseteq\bot}$	$J^4_{E\sqsubseteq\bot}$	$J^5_{E\sqsubseteq\bot}$	$J^6_{E\sqcup C\sqsubseteq\bot}$	$J^7_{E\sqcup C\sqsubseteq\bot}$	$\mathcal{T}_\Delta$	$\mathcal{U}_0$	$\mathcal{U}_1$	$\mathcal{U}_2$
0	Θ	τ_0 τ_1	τ_0 τ_1 τ_2 τ_3	τ_2 τ_4	τ_0 τ_1 τ_2 τ_3 τ_7	τ_2 τ_4 τ_7	τ_0 τ_1 τ_2 τ_3 τ_5 τ_7	τ_0 τ_1 τ_2 τ_3 τ_4 τ_5 τ_7				
	$\bar{\Gamma}$	τ_2 τ_3	τ_7	τ_7	τ_5	τ_5	τ_8	τ_8				
	Θ	$\cancel{\tau_0}$ $\cancel{\tau_1}$		τ_2 τ_4		τ_2 τ_4 τ_7				τ_0 τ_1		
	$\bar{\Gamma}$	τ_2 τ_3		τ_7		τ_5						
1	Θ			$\cancel{\tau_2}$ τ_4						τ_0 τ_1	τ_2 τ_3	
	$\bar{\Gamma}$			τ_7								
	Θ			$\cancel{\tau_2}$ $\cancel{\tau_4}$						τ_0 τ_1	τ_2 τ_3 τ_4	
	$\bar{\Gamma}$			τ_7								
2	Θ									τ_0 τ_1	τ_2 τ_3 τ_4	τ_7
	$\bar{\Gamma}$											
	Θ									τ_0 τ_1	τ_2 τ_3 τ_4	τ_7
	$\bar{\Gamma}$											
Post-processing	Θ									τ_0 τ_1	τ_2 τ_3 τ_4	τ_7 τ_9
	$\bar{\Gamma}$											
	Θ								τ_5 τ_6 τ_8	τ_0 τ_1	τ_2 τ_3 τ_4	τ_7 τ_9
	$\bar{\Gamma}$											

Table 5.1: The single steps of Algorithm 1 applied to Example 10 using the All-Defaults-Splitting-Δ-Operator. Each row represents a step. In columns 3-9 the current contents of the root JUC are displayed whereas in columns 10-13 the contents of the resulting Multi-View-TBox' components are given. Axioms chosen for the Universal TBox $\mathcal{T}_\Delta$ or one of the Aspect TBoxes $\mathcal{U}_n$ in the current step are marked bold. Axioms removed from the splitting sets are crossed out.

n		$J^1_{C\sqsubseteq\bot}$	$J^2_{D\sqcap C\sqsubseteq\bot}$	$J^3_{D\sqcap C\sqsubseteq\bot}$	$J^4_{E\sqsubseteq\bot}$	$J^5_{E\sqsubseteq\bot}$	$J^6_{E\sqcup C\sqsubseteq\bot}$	$J^7_{E\sqcup C\sqsubseteq\bot}$	$\mathcal{T}_\Delta$	$\mathcal{U}_0$	$\mathcal{U}_1$	$\mathcal{U}_2$
0	Θ	τ_0 τ_1	τ_0 τ_1 τ_2 τ_3	τ_2 τ_4	τ_0 τ_1 τ_2 τ_3 τ_7	τ_2 τ_4 τ_7	τ_0 τ_1 τ_2 τ_3 τ_5 τ_7	τ_0 τ_1 τ_2 τ_3 τ_4 τ_5 τ_7				
	$\bar{\Gamma}$	τ_2 τ_3	τ_7	τ_7	τ_5	τ_5	τ_8	τ_8				
	Θ	$\cancel{\tau_0}$ $\cancel{\tau_1}$		τ_2 τ_4		τ_2 τ_4 τ_7				τ_1		
	$\bar{\Gamma}$	τ_2 τ_3		τ_7		τ_5						
1	Θ			τ_2 τ_4		τ_2 τ_4 τ_7				τ_1	τ_3	
	$\bar{\Gamma}$			τ_7		τ_5						
	Θ			$\cancel{\tau_2}$ $\cancel{\tau_4}$						τ_1	τ_2 τ_3	
	$\bar{\Gamma}$			τ_7								
2	Θ									τ_1	τ_2 τ_3	τ_7
	$\bar{\Gamma}$											
	Θ									τ_1	τ_2 τ_3	τ_7
	$\bar{\Gamma}$											
Post-processing	Θ									τ_1	τ_2 τ_3	τ_7 τ_9
	$\bar{\Gamma}$											
	Θ								τ_0 τ_4 τ_5 τ_6 τ_8	τ_1	τ_2 τ_3	τ_7 τ_9
	$\bar{\Gamma}$											

Table 5.2: The single steps of Algorithm 1 applied to Example 10 using the Minimal-Defaults-Splitting-Δ-Operator. Each row represents a step. In columns 3-9 the current contents of the root JUCs are displayed whereas in columns 10-13 the contents of the resulting Multi-View-TBox' components are given. Axioms chosen for the Universal TBox $\mathcal{T}_\Delta$ or one of the Aspect TBoxes $\mathcal{U}_n$ in the current step are marked bold. Axioms removed from the splitting sets are crossed out.

n		$J^1_{C\sqsubseteq\bot}$	$J^2_{D\sqcap C\sqsubseteq\bot}$	$J^3_{D\sqcap C\sqsubseteq\bot}$	$J^4_{E\sqsubseteq\bot}$	$J^5_{E\sqsubseteq\bot}$	$J^6_{E\sqcup C\sqsubseteq\bot}$	$J^7_{E\sqcup C\sqsubseteq\bot}$	$\mathcal{T}_\Delta$	$\mathcal{U}_0$	$\mathcal{U}_1$	$\mathcal{U}_2$
0	Θ	τ_0 τ_1	τ_0 τ_1 τ_2 τ_3	τ_2 τ_4	τ_0 τ_1 τ_2 τ_3 τ_7	τ_2 τ_4 τ_7	τ_0 τ_1 τ_2 τ_3 τ_5 τ_7	τ_0 τ_1 τ_2 τ_3 τ_4 τ_5 τ_7				
	Γ	τ_2 τ_3	τ_7	τ_7	τ_5	τ_5	τ_8	τ_8				
	Θ	$\cancel{\tau_0}$ $\cancel{\tau_1}$		τ_2 τ_4		τ_2 τ_4 τ_7				τ_1		
	Γ	τ_2 τ_3		τ_7		τ_5						
1	Θ			$\cancel{\tau_2}$ τ_4						τ_1	τ_2	
	Γ			τ_7								
	Θ			$\cancel{\tau_2}$ $\cancel{\tau_4}$						τ_1	τ_2	
	Γ			τ_7								
2	Θ									τ_1	τ_2	τ_7
	Γ											
	Θ									τ_1	τ_2	τ_7
	Γ											
Post-processing	Θ									τ_1	τ_2	τ_7 τ_9
	Γ											
	Θ								τ_0 τ_3 τ_4 τ_5 τ_6 τ_8	τ_1	τ_2	τ_7 τ_9
	Γ											

Table 5.3: The single steps of Algorithm 1 applied to Example 10 using the Minimal-Defaults-Splitting-Δ-Operator. Each row represents a step. In columns 3-9 the current contents of the root JUCs are displayed whereas in columns 10-13 the contents of the resulting Multi-View-TBox' components are given. Axioms chosen for the Universal TBox $\mathcal{T}_\Delta$ or one of the Aspect TBoxes $\mathcal{U}_n$ in the current step are marked bold. Axioms removed from the splitting sets are crossed out.

Defaults to Minimal Partitions

Even if we were able solve conflicts using Default Logics, an entailment relation like Lehmann's Lexicographical Entailment is not desirable or needed all the time. For example, using the Minimal-Default Splitting Δ-Operator from Section 5.1.2 we put the minimal number of axioms into

- the Aspect TBoxes $\mathcal{U}_0 = \{\tau_1\}, \mathcal{U}_1 = \{\tau_2\}, \mathcal{U}_2 = \{\tau_7, \tau_9\}$ and
- the Universal TBox with $\mathcal{T}_\Delta = \{\tau_0, \tau_3, \tau_4, \tau_5, \tau_6, \tau_8\}$.

Do there exist alternative consistent Multi-View TBoxes? In Figure 5.1 we give an example for one possible consistent Multi-View TBox that can be obtained from the Default TBox. Effectively, the Multi-View TBox $\mathcal{U}'_0 = \{\tau_1, \tau_7\}, \mathcal{U}'_1 = \{\tau_2\}$ with the same Universal TBox $\mathcal{T}_\Delta$ is consistent well. Yet, it does not fulfill the properties required by Default Logics 2.5.

From the first and second approach towards the third

Considering the number of invalidated entailments, we remember from Section 3.4.3 that the deductive closure of the Multi-View TBox does not contain those entailments that depend on two axioms that happen to be in two different Aspect TBoxes. For the Default Logics solution, we have $|\mathcal{U}_0| \cdot |\mathcal{U}_1| + |\mathcal{U}_0| \cdot |\mathcal{U}_2| + |\mathcal{U}_1| \cdot |\mathcal{U}_2| = 5$ such pairs of axioms that might cause invalidation of entailments. For the alternative solution there exist $|\mathcal{U}'_0| \cdot |\mathcal{U}'_1| = 4$ such invalidating axiom pairs. It is even more interesting that the set of invalidating axiom pairs of the Default Logics solution is a superset of those of the alternative solution. We will refer to this again in Chapter 9.

Consistency

We will now show that any consistent Default TBox can be transformed into a consistent Multi-View TBox with two partitions only.

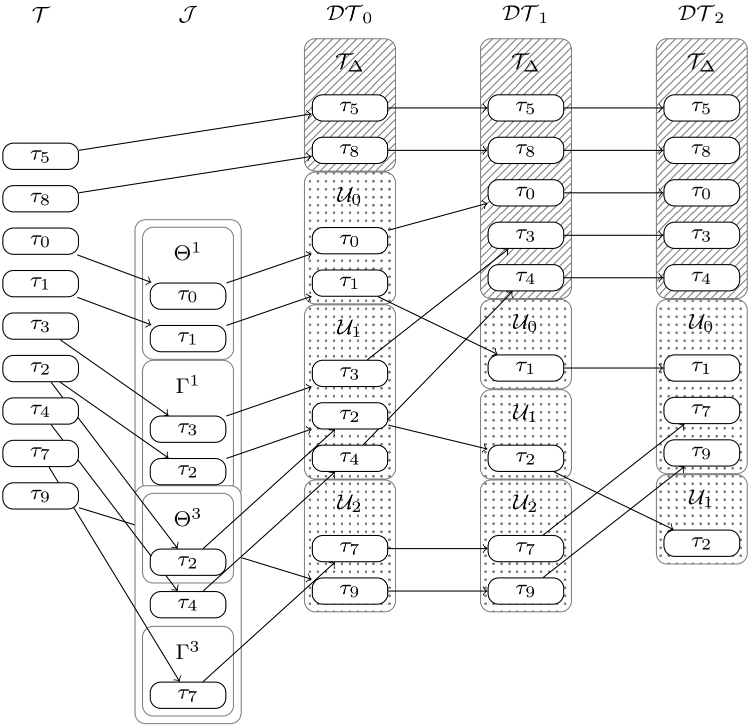


Figure 5.1: Comparison of the Splitting-Δ-Operators.
Transformation of the TBox of Example 10 into a Multi-View-TBox $\mathcal{DT}_0$ using the All-Defaults-Splitting-Δ-Operator. Furthermore how $\mathcal{DT}_0$ compares to a possible solution for the Minimal-Defaults-Splitting-Δ-Operator $\mathcal{DT}_1$ and transformation of $\mathcal{DT}_1$ into the Multi-View-TBox $\mathcal{DT}_2$ according to the Minimal-Partitions-Splitting-Δ-Operator. The actual choice is performed according to the results presented in Table 5.3. On the left-most column, the axioms of the TBox $\mathcal{T}$ are shown. Their relation to other sets are indicated by arrows. They are connected to the second column stating the contents of the root JUCs. The remaining columns show the contents of the parts of the Multi-View-TBoxes $\mathcal{DT}_0$, $\mathcal{DT}_1$, and $\mathcal{DT}_2$. Note that the transition arrows could also go directly from $\mathcal{T}$ and the root JUCs to $\mathcal{DT}_1$ and $\mathcal{DT}_2$.

Theorem 7. *Let $\mathcal{DT} = (\mathcal{T}_\Delta \cup \mathcal{U}_0, \ldots, \mathcal{T}_\Delta \cup \mathcal{U}_N)$ be a consistent Default TBox. Let $\mathcal{U}'_0 = \bigcup_{(n \mod 2)=0} \mathcal{U}_n$ and $\mathcal{U}'_1 = \bigcup_{(n \mod 2)=1} \mathcal{U}_n$. The Multi-View TBox $\mathcal{DT} = (\mathcal{T}_\Delta \cup \mathcal{U}'_0, \mathcal{T}_\Delta \cup \mathcal{U}'_1)$ is consistent.*

Proof. We know from Lemma 2 that for a TBox $\mathcal{T}$ that does not contain axioms with logical contradictions, every JUC w.r.t. $\mathcal{T}$ has at least two axioms. □

Example

The oracle for the Minimal-Partitions-Δ-Operator can be defined similar to the oracle of the Minimal-Defaults-Δ-Operator from Section 5.1.2.

$$\left| \quad chooseThetaAxiom(\mathcal{U}_n, \mathcal{T}_\Delta, \Theta^k_{U \sqsubseteq \bot})\right| = \begin{cases} rand(\Theta^k_{U \sqsubseteq \bot}) & if \quad n = 0 \\ \emptyset & else \end{cases}$$

$$\left| chooseGammaAxiom(\mathcal{U}_n, \mathcal{T}_\Delta, \Gamma^k_{U \sqsubseteq \bot})\right| = \begin{cases} rand(\Gamma^k_{U \sqsubseteq \bot}) & if \quad n > 0 \\ \emptyset & else \end{cases}$$

$$updateIndex(n) = (n + 1) \quad mod \quad 2$$

The only difference lies in the update of the index function. Instead of always creating a fresh partition, we only use the two partitions $\mathcal{U}_0$ and $\mathcal{U}_1$ alternately. We can hence re-use the example from Section 5.1.2, but now with a predefined number of two partitions. The results are given in Table 5.4.

5.2.1 Arbitrary Minimal-Partitions

Giving up the semantics of Default Logics, we can give up the strict requirements (no direct contradictions, acyclic justifications) and, even more, obtain a consistent Multi-View TBox in any case. Furthermore, this Multi-View TBox can be shown to only have exactly two Aspect TBoxes:

Theorem 8. *Let $\mathcal{T}$ be a TBox that contains unsatisfiable concepts. There exists a splitting-Δ-operator for two aspect TBoxes $\mathcal{U}_0, \mathcal{U}_1$ such that the resulting Multi-View-TBox is consistent.*

The proof for Theorem 8 can be found in Appendix B.4. Here, we will illustrate how the two Aspect TBoxes are constructed. The main idea is to successively add exactly one axiom of each $\Theta_{U \sqsubseteq \bot}$ to $\mathcal{U}_0$ and exactly one axiom of each $\Gamma_{U \sqsubseteq \bot}$ to $\mathcal{U}_1$. The remaining axioms are then added to $\mathcal{T}_\Delta$.

5.3 Iterative Splitting-Δ-Operator

Computing JUCs on demand.

Computing all root JUCs for all concepts U that are unsatisfiable w.r.t. a TBox $\mathcal{T}$ is a very expensive task. In contrast to that, computing just one justification for each entailment $\mathcal{T} \models U \sqsubseteq \bot$ can be done alongside computing the entailment. We can hence apply the splitting Δ-operator to the set of the single JUC we found so far. For a Default TBox $\mathcal{DT}_\nu$ that results from this procedure, we bear the risk to not have resolved all root JUCs. As a consequence, we have to refine the solution $\mathcal{DT}_\nu$ by computing additional justifications for the concepts that are still unsatisfiable.

5.3.1 Initial Solution

To obtain an initial solution for the Splitting Δ-operator applied to a TBox $\mathcal{T}$, we compute exactly one JUC for each concept U that is unsatisfiable w.r.t. $\mathcal{T}$. We define root JUCs as before, but on a subset $\mathcal{J}_0$ of the set of all possible JUCs $\mathcal{J}_0 \subseteq \mathcal{J}$. However, we should be aware of the fact that a concept whose unsatisfiability was derived w.r.t. $\mathcal{J}$ may become a root JUC w.r.t. the reduced set of JUCs $\mathcal{J}_0$. This is illustrated in Example 14.

If we now apply the Splitting Δ-operator to $\mathcal{J}_0$ we obtain an initial solution $\mathcal{DT}_0 = (\mathcal{T}_\Delta \cup \mathcal{U}_0, \ldots, \mathcal{T}_\Delta \cup \mathcal{U}_N)$. Yet, there may exist concepts $\tilde{U}$ that are still unsatisfiable w.r.t. $\mathcal{T}_\Delta$ alone or w.r.t.

n		$J^1_{C\sqsubseteq\bot}$	$J^2_{D\sqcap C\sqsubseteq\bot}$	$J^3_{D\sqcap C\sqsubseteq\bot}$	$J^4_{E\sqsubseteq\bot}$	$J^5_{E\sqsubseteq\bot}$	$J^6_{E\sqcup C\sqsubseteq\bot}$	$J^7_{E\sqcup C\sqsubseteq\bot}$	$\mathcal{T}_\Delta$	$\mathcal{U}_0$	$\mathcal{U}_1$	
$n=0$	Θ	τ_0 τ_1	τ_0 τ_1 τ_2 τ_3	τ_2 τ_4	τ_0 τ_1 τ_2 τ_3 τ_7	τ_2 τ_4 τ_7	τ_0 τ_1 τ_2 τ_3 τ_5 τ_7	τ_0 τ_1 τ_2 τ_3 τ_4 τ_5 τ_7				
	$\bar{\Gamma}$	τ_2 τ_3	τ_7	τ_7	τ_5	τ_5	τ_8	τ_8				
	Θ	$\cancel{\tau_0}$ $\cancel{\tau_1}$		τ_2 τ_4		τ_2 τ_4 τ_7				τ_1		
	$\bar{\Gamma}$	τ_2 τ_3		τ_7		τ_5						
$n=1$	Θ			$\cancel{\tau_2}$ τ_4						τ_1	τ_2	
	$\bar{\Gamma}$			τ_7								
	Θ			$\cancel{\tau_2}$ $\cancel{\tau_4}$						τ_1	τ_2	
	$\bar{\Gamma}$			τ_7								
$n=2$	Θ									τ_1	τ_2 τ_7	
	$\bar{\Gamma}$											
	Θ									τ_1	τ_2 τ_7	
	$\bar{\Gamma}$											
Post-processing	Θ									τ_1	τ_2 τ_7 τ_9	
	$\bar{\Gamma}$											
	Θ								τ_0 τ_3 τ_4 τ_5 τ_6 τ_8	τ_1	τ_2 τ_7 τ_9	
	$\bar{\Gamma}$											

Table 5.4: The single steps of Algorithm 1 applied to Example 10 using the a splitting-Δ-operator for the computation of minimal partitions. Each row represents a step. In columns 2-6 the current contents of the root JUCs are displayed whereas in columns 7-10 the contents of the resulting Multi-View-TBox' components are given. Axioms chosen for the Universal TBox $\mathcal{T}_\Delta$ or one of the Aspect TBoxes $\mathcal{U}_n$ in the current step are marked bold. Axioms removed from the splitting sets are crossed out.

$\mathcal{T}_\Delta \cup \mathcal{U}_n$ for at least one $0 \leq n \leq N$[1]. For each of these $\tilde{U}$ we have to compute one new JUC $J^{k_1}_{\tilde{U}\sqsubseteq\bot}$ that does not occur in $\mathcal{J}_0$. We add these new $k_1, \ldots, K_1$ JUCs to $\mathcal{J}_0$ and perform the splitting again on $\mathcal{J}_1 = \mathcal{J}_0 \bigcup_{k_1}^{K_1} \{J^{k_1}_{\tilde{U}\sqsubseteq\bot}\}$.

5.3.2 General Procedure

In general, we have to compute the set of concepts $\tilde{U}_\nu$ that are unsatisfiable w.r.t. $\mathcal{DT}_{\nu-1}$[2]. For each of these $\tilde{U}_\nu$, we compute one JUC such that $J^{k_\nu}_{\tilde{U}_\nu\sqsubseteq\bot} \cap \mathcal{J}_{\nu-1} = \emptyset$.
We set $\mathcal{J}_\nu = \mathcal{J}_{\nu-1} \bigcup_{k_\nu=0}^{K_\nu} J^{k_\nu}_{\tilde{U}_\nu\sqsubseteq\bot}$ and determine $\mathcal{DT}_\nu$ w.r.t. $\mathcal{J}_\nu$. Since the number of JUCs is finite, the procedure terminates eventually.

5.3.3 Complexity

The number of iterations is bounded by the maximum number of root JUCs for an non-purely derived unsatisfiable concept. We denote this number by r_{max}. In the best case, the procedure finds a solution after the first step. This is the case when for each non purely derived unsatisfiable concept we can find a root JUC and these cover the set of all root JUCs.

We do not have to compute all the dependent JUCs. In the best case, we have to compute as many dependent JUCs as there exist purely derived unsatisfiable concepts. We denote the number of partially derived unsatisfiable concepts by c_{part} and the number of purely derived unsatisfiable concepts by c_{pure}. In the worst case, we have to compute $r_{max} \cdot (c_{part} + c_{pure})$ derived JUCs. Computing the iterative solution, we may benefit from not having to compute all JUCs, as a result.

[1] There may exist $m \neq n$ such that both $\mathcal{T}_\Delta \cup \mathcal{U}_n \models U \sqsubseteq \bot$ and $\mathcal{T}_\Delta \cup \mathcal{U}_m \models U \sqsubseteq \bot$, as well as $\mathcal{T}_\Delta \models U \sqsubseteq \bot$ at the same time.

[2] We start with $\mathcal{DT}_0 = \mathcal{T}$

5.3.4 Dependencies

Adding new JUCs to $\mathcal{J}'$ may change the dependencies of the JUCs within $\mathcal{J}'$. Assume we add some JUC $J_{U_i \sqsubseteq \bot}$ to $\mathcal{J}'$ that are root JUCs w.r.t $\mathcal{J}$. Consider all JUCs $J_{U_j \sqsubseteq \bot}$ that were root JUCs w.r.t. $\mathcal{J}'$ before adding new JUCs to $\mathcal{J}'$. If for some $J_{U_j \sqsubseteq \bot}$ we have that $J_{U_j \sqsubseteq \bot} \supset J_{U_i \sqsubseteq \bot}$, then those $J_{U_j \sqsubseteq \bot}$ become dependent JUCs w.r.t. $\mathcal{J}'$ after adding new JUCs.

Assume we add some JUC $J_{U_i \sqsubseteq \bot}$ to $\mathcal{J}'$ that is not a root JUC w.r.t $\mathcal{J}$. Then all JUCs that were root JUCs w.r.t. $\mathcal{J}'$ before the adding remain to be a root unsat JUC (and not a non-root JUC) w.r.t. $\mathcal{J}'$ after the adding.

Example 14. *Assume the TBox $\mathcal{T} = \{B \sqsubseteq A, C \sqsubseteq B, D \sqsubseteq C, D \sqsubseteq \neg A, B \sqsubseteq \neg F, D \sqsubseteq F, E \sqsubseteq D\}$. Obviously, the concepts D and E are unsatisfiable w.r.t. $\mathcal{T}$ with the justifications:*

$$\begin{array}{lcccc}
J^1_{\mathcal{T},D\sqsubseteq\bot} & = & \overbrace{\{\ B \sqsubseteq A, C \sqsubseteq B\ \}}^{\Theta} & \cup & \overbrace{\{D \sqsubseteq C, D \sqsubseteq \neg A\}}^{\Gamma} \\
J^2_{\mathcal{T},D\sqsubseteq\bot} & = & \{B \sqsubseteq \neg F, C \sqsubseteq B\} & \cup & \{\ D \sqsubseteq C, D \sqsubseteq F\ \} \\
J^3_{\mathcal{T},E\sqsubseteq\bot} & = & J^1_{\mathcal{T},D\sqsubseteq\bot} & \cup & \{\quad E \sqsubseteq D \quad\}
\end{array}$$

The only root JUCs are $J^1_{\mathcal{T},D\sqsubseteq\bot}$ and $J^2_{\mathcal{T},D\sqsubseteq\bot}$. Assume we find $\mathcal{J}_0 = \{J^2_{\mathcal{T},D\sqsubseteq\bot}, J^3_{\mathcal{T},E\sqsubseteq\bot}\}$ as the first JUCs for C and D. So assume that $\mathcal{DT}_1$ results from applying some Splitting-Δ-Operator. Depending on the Splitting-Δ-Operator, the resulting Default TBox $\mathcal{DT}_0$ can entail $D \sqsubseteq \bot$, but does not have to.

Assume that applying a Splitting-Δ-Operator $\mathcal{U}_0$ contains the axiom $C \sqsubseteq B$ and $\mathcal{U}_1$ contains $D \sqsubseteq C$. This is, for example, the case for the All-Defaults-Splitting-Δ-Operator. Then for each root JUC one axiom is already contained in one of the partitions. Hence $\mathcal{DT}_0$ is consistent. Both root JUCs mask each other, so we are able to solve both by solving just one. Note that we do not even compute $J^1_{\mathcal{T},D\sqsubseteq\bot}$ in that case.

Assume now that we applied a Splitting-Δ-Operator such that neither $\mathcal{U}_0$ contains the axiom $C \sqsubseteq B$ nor $\mathcal{U}_1$ contains $D \sqsubseteq C$. This can be, for example, the case when applying the Minimal-Defaults-Splitting-

Δ-Operator to $\mathcal{T}$. We find that none of the axioms of $J^1_{\mathcal{T},D\sqsubseteq\bot}$ is contained in any partition. Hence, $J^1_{\mathcal{T},D\sqsubseteq\bot} \subseteq \mathcal{T}_\Delta$ and $\mathcal{T}_\Delta \models D \sqsubseteq \bot$. Since $\mathcal{DT}_0 \not\models E \sqsubseteq \bot$, we just have to compute another JUC for D. We obtain $\mathcal{J}_1 = \mathcal{J}_0 \cup \{J^1_{\mathcal{T},C\sqsubseteq\bot}\}$. We now update the partitions and get a consistent $\mathcal{DT}_1$.

Since we compute possibly not all JUCs, the Unsat-Splitting cannot be used to determine the candidates for the n-th partition $\mathcal{U}_n$. We might not be able to determine concepts that are unsatisfiable w.r.t $\mathcal{T} \setminus \bigcup_{m=0}^{n-1} \mathcal{U}_m$ anymore. This is illustrated by Example 15.

Example 15. *Assume that we extended the TBox of Example 14 by the following axioms: $\mathcal{T} = \mathcal{T} \cup \{E \sqsubseteq B, E \sqsubseteq F, G \sqsubseteq E\}$. Obviously, the concept G is unsatisfiable w.r.t. $\mathcal{T}$. Besides the JUCs from Example 14 we have some additional JUCs:*

$$\begin{array}{lcccc}
 & & \overbrace{\hphantom{\{B \sqsubseteq \neg F, C \sqsubseteq B, D \sqsubseteq C\}}}^{\Theta} & & \overbrace{\hphantom{\{E \sqsubseteq D, E \sqsubseteq F\}}}^{\Gamma} \\
J^4_{\mathcal{T},E\sqsubseteq\bot} & = & \{B \sqsubseteq \neg F, C \sqsubseteq B, D \sqsubseteq C\} & \cup & \{E \sqsubseteq D, E \sqsubseteq F\} \\
J^5_{\mathcal{T},E\sqsubseteq\bot} & = & \{\qquad B \sqsubseteq \neg F \qquad\} & \cup & \{E \sqsubseteq B, E \sqsubseteq F\} \\
J^6_{\mathcal{T},G\sqsubseteq\bot} & = & J^3_{\mathcal{T},E\sqsubseteq\bot} & \cup & \{\quad G \sqsubseteq E \quad\} \\
J^7_{\mathcal{T},G\sqsubseteq\bot} & = & J^4_{\mathcal{T},E\sqsubseteq\bot} & \cup & \{\quad G \sqsubseteq E \quad\} \\
J^8_{\mathcal{T},G\sqsubseteq\bot} & = & J^5_{\mathcal{T},E\sqsubseteq\bot} & \cup & \{\quad G \sqsubseteq E \quad\}
\end{array}$$

The root JUCs are $J^1_{\mathcal{T},D\sqsubseteq\bot}$, $J^2_{\mathcal{T},D\sqsubseteq\bot}$, $J^4_{\mathcal{T},E\sqsubseteq\bot}$, and $J^5_{\mathcal{T},E\sqsubseteq\bot}$.

Assume we find $\mathcal{J}_0 = \{J^2_{\mathcal{T},D\sqsubseteq\bot}, J^3_{\mathcal{T},E\sqsubseteq\bot}, J^5_{\mathcal{T},E\sqsubseteq\bot}, J^6_{\mathcal{T},G\sqsubseteq\bot}\}$ as the first JUCs for D, E and G. We further assume that $\mathcal{DT}_1$ results from applying some Splitting-Δ-Operator. The splitting operator does not determine the unsatisfiability of concepts correctly w.r.t. $\mathcal{J}_0$

Axiom $E \sqsubseteq D$ is not contained in any justification in $\mathcal{J}_0$. It is contained in $\Theta^6_{\mathcal{T},G\sqsubseteq\bot}$, but not part of any Γ-set w.r.t. $\mathcal{J}_0$. Hence, it can be contained in $\mathcal{U}_0$. On the other hand, $E \sqsubseteq D$ is contained in a Γ-set w.r.t. $\mathcal{J}$. Indeed, $E \sqsubseteq D \in \Gamma^3_{\mathcal{T},E\sqsubseteq\bot}$, but we have not yet even *computed the corresponding JUC $J^3_{\mathcal{T},E\sqsubseteq\bot}$. Furthermore, it is possible that $\mathcal{DT}_0 \models E \sqsubseteq \bot$, because we may have not solved the root JUC $J^4_{\mathcal{T},E\sqsubseteq\bot}$.*

Example 15 shows that the iterative procedure can provide us with partitions whose contents must be changed in the iterations that follow. As a consequence, we have to possibly recompute the partitions in each iteration. The whole procedure is illustrated by Algorithm 5. We have to determine and manage information for those concepts that are unsatisfiable w.r.t. $\mathcal{T}\backslash\bigcup_{m=0}^{n-1}\mathcal{U}_m$ when computing the partitions. The Unsat-Splitting on the incomplete set of JUC $\mathcal{J}$ cannot provide this information anymore.

5.4 Solutions In Between

Instead of going to the extremes, i.e. either adding all or just one axiom from each of the splitting sets to the Aspect TBoxes, other choices are possible as well. Even cyclic justifications can be solved by transforming the incoherent TBox $\mathcal{T}$ to a Multi-View TBox $\mathcal{DT}$. Yet, it cannot be guaranteed that $\mathcal{DT}$ is consistent. What makes the approach attractive then?

In the case of acyclic incoherent TBoxes it may be desirable to prevent certain entailments from being invalidated. We can start with a solution from one of the approaches above and *refine* it to meet some pre-defined constraints. Assume that, in Example 2, we had the constraint to preserve the entailment that penguins are flying animals, i. e. $Penguin \sqsubseteq FlyingAnimal$. We cannot add this as an axiom, because this would lead to a direct contradiction. However, the Multi-View TBox with the following Aspect TBoxes is consistent:

$$\begin{aligned}\mathcal{U}_0 &= \{Penguin \sqsubseteq Bird, Bird \sqsubseteq FlyingAnimal\}\\ \mathcal{U}_1 &= \{Penguin \sqsubseteq \neg FlyingAnimal\}\end{aligned}$$

Algorithm 5 Iterative Splitting-Δ-Operator.

Function:
$\mathcal{DT} \leftarrow computeMultiViewTBoxIterative($
$\mathcal{T}, \Theta^0_{U\sqsubseteq\bot}, \Gamma^0_{U\sqsubseteq\bot} \ldots, \Theta^K_{U\sqsubseteq\bot}, \Gamma^K_{U\sqsubseteq\bot}, \Delta)$

Input:
1. TBox $\mathcal{T}$
2. Splitting sets $\Theta^0_{U\sqsubseteq\bot}, \Gamma^0_{U\sqsubseteq\bot} \ldots, \Theta^K_{U\sqsubseteq\bot}, \Gamma^K_{U\sqsubseteq\bot})$
3. Δ-Operator

Output:
$\mathcal{DT}$: Consistent Default TBox ,
$\mathcal{J}$: Dependency graph for JUCs

```
$\mathcal{DT}_0 \leftarrow \langle\mathcal{T}\rangle$    $\mathcal{J}_{-1} \leftarrow \emptyset$    $\nu = 0$

while $\mathcal{DT}_\nu$ is not consistent do
    // Take all JUCs computed so far.
    $\mathcal{J}_{\nu+1} \leftarrow \mathcal{J}_\nu$
    // Compute all concepts that are unsatisfiable
    // w.r.t. the current Default TBox.
    $\Omega \leftarrow \{U \mid U \in sig(\mathcal{DT}_\nu) \wedge \mathcal{DT}_\nu \models U \sqsubseteq \bot\}$

    for $U \in \Omega$ do
        // Compute one new justification for each concept
        // that is unsatisfiable w.r.t. the current Default TBox.
        $\mathcal{J}_\nu \leftarrow \mathcal{J}_\nu \cup \{J_{\mathcal{DT}\models U\sqsubseteq\bot}\}$ such that
          $\mathcal{J}_{\nu-1} \cap \{J_{\mathcal{DT}\models U\sqsubseteq\bot}\} = \emptyset$
    end for

    // Compute new Default TBox by applying the Δ-operator to
    // all the already computed JUCs.
    $\mathcal{DT}_{\nu+1} \leftarrow \Delta(\bigcup_{J_{\mathcal{DT}_\nu\models U\sqsubseteq\bot} \in \mathcal{J}_\nu} J_{\mathcal{DT}_\nu\models U\sqsubseteq\bot})$
    // Update new Default TBox by adding all those axioms to $\mathcal{T}_\Delta$
    // that do not occur in any of the already computed JUCs.
    $\mathcal{T}_{\Delta_{\nu+1}} \leftarrow \mathcal{T}_{\Delta_{\nu+1}} \cup \left(\mathcal{T} \setminus \bigcup_{J_{\mathcal{DT}_\nu \models U \sqsubseteq \bot} \in \mathcal{J}_\nu} J_{\mathcal{DT}_\nu \models U \sqsubseteq \bot}\right)$

    $\nu \leftarrow \nu + 1$
end while

// Set the last Default TBox and all the computed
// JUCs as the result.
$\mathcal{DT} \leftarrow \mathcal{DT}_{\nu-1} \quad \mathcal{J} \leftarrow \mathcal{J}_{\nu-1}$
```

It is interesting to note that we, indeed, transformed *all* axioms of the root JUCs, yet we cannot apply Default Logics entailment.

A proposal for an implementation of this approach is made by Algorithm 6. It makes non-deterministic choices that might result in inconsistent Multi-View TBoxes. Nevertheless, once the root JUCs are known this can be done by a series of set operations, i.e. checking that at least two axioms of each root JUC are contained in either $\mathcal{U}_0$ or $\mathcal{U}_1$. Because it is unclear what strategy the choice shall follow, we did, however, not investigate this algorithm further and leave it for future work.

5.5 The Splitting-Δ-operator on Lattices

For DL TBoxes, lattices can be used to define monotone properties on sets of axioms such as concept subsumption hierarchy (i.e. TBox classification) or concept unsatisfiability. In the following we show how lattices can be used to encode solutions to the Unsat-Splitting Δ-Operator.

5.5.1 Default TBox Lattice

We can use lattices to encode the partitions of a Default TBox $\mathcal{T}$. We know from Section 2.2 that the set of all monotone Boolean functions over a finite set of propositional variables P encodes a distributive lattice $\mathbb{B}^{\mathcal{T}}$. According to [Peñaloza, 2009] we can build a homomorphism that transforms $\mathbb{B}^{\mathcal{T}}$ to another lattice $\mathbb{L}$ where we can define weights α on axioms. Those weights are specified by a labeling function $lab : \mathcal{T} \to \mathbb{Z}$. The homomorphism can be defined in such a way that the property for the pinpointing formula is preserved on the lattice $\mathbb{L}$. Furthermore, we can define a border on the lattice such that certain properties follow only from axioms that have a weight greater or than a given value for these properties.

Assume the Default TBox $\mathcal{DT}$ that originates from $\mathcal{T}$ and consists of the partitions $\mathcal{U}_0, \ldots, \mathcal{U}_N$ and the Universal TBox $\mathcal{T}_\Delta$. We can define the following set of properties:

$$\mathcal{P}_n = \left\{ B \sqsubseteq A \in \mathcal{T} \setminus \bigcup_{m=0}^{n-1} \mathcal{U}_m \quad | \quad \left(\mathcal{T}_\Delta \cup \bigcup_{m=n}^{N} \mathcal{U}_m \right) \models A \sqcap B \right\}$$

In that sense, P_n encodes all those axioms which belong to a partition $\mathcal{U}_n$. We label all axioms in $\mathcal{P}_n$ by n. All axioms $t \in \mathcal{T}_\Delta$ are assigned the label $lab(t) = -1$. By considering only axioms of level n, we effectively have to restrict ourselves to the sub-lattice formed by the axioms from $\mathcal{T} \setminus \bigcup_{m=0}^{n-1} \mathcal{U}_m$. According to the definition of partitions in Lehmann's Default Logics (cf Section 2.5), the minimal set of axioms that satisfy this property are the axioms in $\mathcal{U}_n \cup \mathcal{T}_\Delta$. Furthermore, this property is still satisfied by adding axioms $t^{>n}$ with a label greater than n, although the axioms $t^{>n}$ do not fulfill the property:

$$(\mathcal{T}_1, \mathcal{I}) \in \mathcal{P}_n \Rightarrow \forall \mathcal{T}_2 \supset \mathcal{T}_1 : (\mathcal{T}_2, \mathcal{I}) \in \mathcal{P}_n$$

Hence each property $\mathcal{P}_n$ is a monotone property w.r.t. $(\mathcal{T} \setminus \bigcup_{m=0}^{n-1} \mathcal{U}_m, \mathcal{I})$. As stated in Peñaloza [2009], once we computed the pinpointing formula for all unsatisfiable concepts, then computing different partitions can be restricted to finding another labeling function lab for the axioms in $\mathcal{T}$.

As a result, we can consider the problem of finding a solution for the Minimal Unsat-Splitting Δ-operator for a TBox $\mathcal{T}$ as finding a homomorphism from the lattice $\mathbb{B}^{\mathcal{T}}$ of monotone Boolean formulas over P to a weighed lattice $\mathbb{L}$. Yet, the central challenge is still finding the pinpointing formula—or in other words finding the set of all JUCs.

Algorithm 6 Computing an alternative solution.

Input:

1. Minimal-Split Multi-View TBox $\mathcal{DT} = (\mathcal{U}_0 \cup \mathcal{T}_\Delta, \mathcal{U}_1 \cup \mathcal{T}_\Delta)$,
2. K Root JUCs $J^k_{U \sqsubseteq \bot}$ for $k = 0, \ldots, K$

Output:

Transformed Default TBox $\mathcal{DT}'$

```
repeat
    // Choose Aspect TBox to change
    choose $\mathcal{U}_{change} \in \{\mathcal{U}_0, \mathcal{U}_1\}$

    // Pick up axiom at random
    choose $\alpha \in \mathcal{U}_{change}$

    // Determine root JUCs that contain $\alpha$
    $\mathcal{J} \leftarrow \{J^k_{U \sqsubseteq \bot} | \alpha \in J^k_{U \sqsubseteq \bot}\}$

    // choose one root JUC to change
    choose $J^{k_0}_{U_0 \sqsubseteq \bot} \in \mathcal{J}$

    // choose axiom to exchange ( transforms $\mathcal{DT} \rightarrow \mathcal{DT}'$)
    choose $\alpha$ from $J^{k_0}_{U_0 \sqsubseteq \bot}$

until $\mathcal{DT}'$ is consistent or no more choices possible
```

Part III

Evaluation

6

Finding Optimal Solutions

The Splitting-Δ-Operators with potential minimal loss of implicit information are non-deterministic. Depending on the performance measure there may exist a single global optimum. Yet, there does not exist a general rule for computing such a global optimum—besides the brute-force approach of computing all possible solutions, which is not feasible for large ontologies. Computing some solution is, in contrast, relatively cheap. Hence, we propose to approximate the global optimum by starting with a initial solution which is successively improved.

Note that we do not provide an actual implementation of an optimization strategy such as Simulated Annealing or genetic algorithms. Instead, we investigate whether it would actually make sense to apply such an optimization strategy, in general.

Plan of This Chapter

The formal foundations for choices and solutions are introduced in Section 6.1. In Section 6.2 we introduce an entropy measure for judging the quality of solutions.

6.1 Choices and Solutions

Formalizing the way axioms are chosen.

According to Theorem 4, a Splitting-Δ-operator produces a coherent Multi-View-TBox as a result. In Chapter 5 we defined two of those operators, each of which relies upon a non-deterministic choice function. We have to specify this choice function in order to obtain a solution for those Splitting-Δ-Operators.

6.1.1 Unique Choices for Minimal-Defaults-Splitting-Δ-operator

As can be seen from Algorithm 1, *the choices being made locally influence the overall or global choice*. We hence define the choice function locally and globally.

Definition 21. *Let $\mathcal{T}$ be a TBox and $\mathcal{DT} = (\mathcal{T}_\Delta \cup \mathcal{U}_0, \ldots, \mathcal{T}_\Delta \cup \mathcal{U}_N)$ be a Multi-View TBox. Let $J^k_{U \sqsubseteq \bot}$ be the $k = 0, \ldots, K$ JUCs for $\mathcal{T}$. Then the function $c_k : J^k_{U \sqsubseteq \bot} \to \{\mathcal{T}_\Delta, \mathcal{U}_0, \ldots, \mathcal{U}_N\}$ is called the* local choice *for $\mathcal{DT}$ w.r.t. the JUC $J^k_{U \sqsubseteq \bot}$.*
The family of local choices $(c_k)^K_{k=0}$ defines the global choice *for $\mathcal{DT}$ w.r.t all JUCs for $\mathcal{T}$.*

Neither a local choice nor a global choice happens to be an injective function. Two axioms from $J^k_{U \sqsubseteq \bot}$ may be mapped to the same set, either $\mathcal{T}_\Delta$ or one of $\mathcal{U}_n$. There do not have to exist mappings to all of these sets for a local choice. Hence local choices are not even surjective. In contrast to that, the global choice indeed is surjective, because neither $\mathcal{T}_\Delta$ nor one of $\mathcal{U}_n$ is allowed to be empty but all of them have to contain elements from JUCs. We may define the *inverse local choice* $\bar{c}_k$ (and similarly also the inverse global choice) which tells us from what JUCs some axiom was chosen from. in the sequel, we refer to global choices simply as choices.

The following lemma states that not only a choice is defined by a Multi-View TBox but that, for the Splitting-Δ-Operator, each choice defines a unique solution. This is an important property when we want to apply optimization techniques that require to be able to slightly change the current solution. Solutions are computed by computing choices. If we change the choice a bit, then we require the corresponding solution to be unique.

Unique solutions

Lemma 9. *Let $\mathcal{T}$ be a TBox and $J^k_{U\sqsubseteq\bot}$ be the $k = 0, \dots, K$ JUCs for $\mathcal{T}$. Each Multi-View TBox $\mathcal{DT}$ that is a solution for the Minimal-Splitting Δ-operator corresponds to exactly one choice.*

Proof. Assume there existed two different global choices $(c_k)^K_{k=0}$ and $(d_k)^K_{k=0}$ for a Multi-View TBox $\mathcal{DT}$ resulting from the Minimal-Splitting Δ-Operator applied to some TBox $\mathcal{T}$. Since the choices are different, they must differ in one of their local choices. Let c_k and d_k be one of the local choices in which $(c_k)^K_{k=0}$ and $(d_k)^K_{k=0}$ differ. By definition, c_k maps all axioms of the JUC $J^k_{U\sqsubseteq\bot}$ to either $\mathcal{T}_\Delta$ or one of $\mathcal{U}_n$. Since c_k and d_k differ, there must exist one axiom α such that $c_k(\alpha) \neq d_k(\alpha)$. By definition $\mathcal{T}_\Delta \cup \mathcal{U}_n = \emptyset$ for all $\mathcal{U}_n$.

The only possibility for which $c_k(\alpha) \neq d_k(\alpha)$ is $c_k(\alpha) = \mathcal{U}_m$ and $d_k(\alpha) = \mathcal{U}_n$ or vice versa for $m \neq n$. By definition $\mathcal{U}_n \neq \mathcal{U}_m$ for $m \neq n$, but $\alpha \in \mathcal{U}_m$ and $\alpha \in \mathcal{U}_n$ leading to a contradiction. Hence, *every solution for the Minimal-Splitting-Δ-Operator uniquely corresponds to exactly one choice and vice versa.* □

6.1.2 Mutating Choices for the Minimal-Defaults-Splitting-Δ-operator

If we computed one solution $\mathcal{DT}_0$, is it possible to change this solution a little bit such that the result $\mathcal{DT}_1$ is a valid solution, as well? In other words, given choice $(c_k)^K_{k=0}$, and changing the local choice c_k, what is the resulting global choice $(d_k)^K_{k=0}$?

For considering this question, it is helpful to think of the space of all possible choices as a tree with edges e and nodes v. Every edge e_i stands for a set of choices c_k and no choice must occur more than once on a path. The collection of choices on any path from the root to a leaf node must contain all local choices for $k = 0, \ldots, K$. A node v can be interpreted in two different ways:

(i) If v is the starting node of an edge e, then v consists of the Θ-sets of the root JUCs $J^k_{U \sqsubseteq \perp}$ that correspond the to choices c_k that are aligned to the edge.

(ii) If v is the ending node of an edge e, then v consists of the Γ-sets of the root JUCs $J^k_{U \sqsubseteq \perp}$ that correspond the to choices c_k that are aligned to the edge.

Figure 6.1 illustrates the possible choices for Example 10.

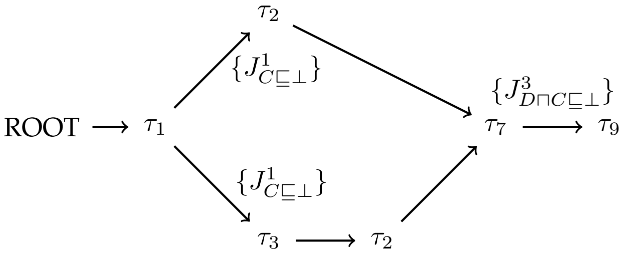
For mutating a local choice c_{k_i} for a global choice $(c_k)_{k=0}^K$ we first determine the starting node of the edge c_k occurs in. From this node we backtrack towards the root until we find a node v which has more than one child node, i.e. more than one possibility for making local choices. In case v is the root node, we cannot make another choice. Otherwise, we choose one of the child nodes of v and choose a path to a leaf node. We, thereby, must take into account that we change more local choices than only the local choice c_k. The procedure is, however, also non-deterministic but leads to a valid solution.

How to compute new states from old ones.

6.1.3 Mutating Arbitrary Choices

As could be seen in Section 5.4, there exist valid solutions besides the ones produced by the Minimal-Defaults-Splitting-Δ-Operator. We may mutate any global choice $(c_k)_{k=0}^K$ that refers to a valid solution to another global choice $(d_k)_{k=0}^K$. However, if we do not enforce the new choice to meet the criteria of the Minimal-Defaults-Splitting-Δ-Operator or the Default Δ-Operator, then we must perform a validity-check for the new global choice. In particular, we have to check whether the requirements for an arbitrary Δ-Operator are still fulfilled (cf Definition 17):

Figure 6.1: The two possible choices for Example 10 represented as tree. The edges are labeled by the set of root JUCs whose local choices are represented by this edge.



For each root JUC two axioms have to be contained in two different Aspect TBoxes, and no root JUC is contained in any of the Aspect TBoxes.

6.2 Qualitative (Default) TBox Assessment

So far we took the number of axioms in the finite deductive closure of a Default TBox for comparing its quality w.r.t. the TBox is was created from. In the presence of an ABox, choosing the Default TBox with the finite deductive closure that preserves most entailments w.r.t. original TBox may be a false friend. Simply counting the number of entailments we do not take into account the *impact of the preserved and lost entailments*.

6.2.1 Conflicts for $\Delta(\mathcal{T})$

Even though we invalidated entailments of the form $U \sqsubseteq \bot$, the axioms that caused the contradiction in the original TBox $\mathcal{T}$ are still present in $\Delta(\mathcal{T})$. Yet, one possible reason for concepts to be entailed unsatisfiable is when $\mathcal{T}$ entails both

$U \sqsubseteq D$ and $U \sqsubseteq \neg D$. For classical entailment these axioms are semantically equivalent to $U \sqsubseteq D \sqcap \neg D$, and hence $U \sqsubseteq \bot$ is a logical consequence for $\mathcal{T}$.

Semantic equivalence under the Splitting

This semantic equivalence is not valid w.r.t. a Splitting-Δ-Operator $\Delta(\mathcal{T})$ and its entailment relationship $\models$. In general, the axioms $D \sqsubseteq C$ and $D \sqsubseteq E$ are not semantically equivalent to $D \sqsubseteq C \sqcap E$. We give an example for this:

Example 16 (Disjunction of axioms in a Multi-View TBox). *Consider the TBox $\mathcal{T} = \{B \sqsubseteq A, C \sqsubseteq B, D \sqsubseteq C, D \sqsubseteq \neg A\}$. Applying the Minimal-Defaults Δ-Operator we obtain four possible Default TBoxes:*

	$\mathcal{T}_\Delta^0$	$\mathcal{U}_0^0$	$\mathcal{U}_1^0$
$\mathcal{DT}^0$	$D \sqsubseteq C, B \sqsubseteq A$	$C \sqsubseteq B$	$D \sqsubseteq \neg A$
$\mathcal{DT}^1$	$D \sqsubseteq \neg A, B \sqsubseteq A$	$C \sqsubseteq B$	$D \sqsubseteq C$
$\mathcal{DT}^2$	$D \sqsubseteq C, C \sqsubseteq B$	$B \sqsubseteq A$	$D \sqsubseteq \neg A$
$\mathcal{DT}^3$	$D \sqsubseteq \neg A, C \sqsubseteq B$	$B \sqsubseteq A$	$D \sqsubseteq C$

According to $\models$, each of these Multi-View TBoxes entails axioms that would refer to a contradiction under classical entailment $\models$. We have that $\mathcal{DT}^0 \models$ both axioms $D \sqsubseteq A$ and $D \sqsubseteq \neg A$. The first is a logical consequence of $\mathcal{T}_\Delta^0 \cup \mathcal{U}_0^0$, the latter is a logical consequence of $\mathcal{T}_\Delta^0 \cup \mathcal{U}_1^0$.[1] For each of the solutions there exists at least one such conflict, yet there exist solutions with even two conflicts:

$$\begin{array}{lcl}
\mathcal{DT}^0 & \models & \{D \sqsubseteq A, D \sqsubseteq \neg A, D \sqsubseteq B, D \sqsubseteq \neg B\} \\
\mathcal{DT}^1 & \models & \{D \sqsubseteq A, D \sqsubseteq \neg A\} \\
\mathcal{DT}^2 & \models & \{D \sqsubseteq C, D \sqsubseteq \neg C\} \\
\mathcal{DT}^3 & \models & \{D \sqsubseteq B, D \sqsubseteq \neg B, D \sqsubseteq C, D \sqsubseteq \neg C\}
\end{array}$$

[1] Note that for a View we defined classical entailment.

Definition 22 (Conflicts)**.** *Let $\mathcal{DT}$ be a consistent Multi-View TBox and U, D be concepts in the signature of $\mathcal{DT}$. We say that $\mathcal{DT}$ contains the conflict $\mathcal{C} = \{U \sqsubseteq D, U \sqsubseteq \neg D\}$, if $\mathcal{DT} \models \mathcal{C}$.*

Minimizing conflicts

Minimizing the number of invalidated entailments may be suboptimal w.r.t. the conflicts that are contained in a solution to the Splitting-Δ-Operator. In Example 16, the finite deductive closure of the Multi-View TBoxes $\mathcal{DT}^0$ and $\mathcal{DT}^3$ contains one axiom (in particular eight axioms for each closure) more than the closures of $\mathcal{DT}^1$ and $\mathcal{DT}^2$ (seven axioms each). On the other hand, each of them contains one conflict more than the other two.

In Example 16, each conflict refers to a concept that is unsatisfiable w.r.t. the original TBox $\mathcal{T}$. While we leave a proof for the general case for future work, we assume that this is the case for any conflict when we apply a Splitting-Δ-Operator $\Delta(\mathcal{T})$.

Since conflicts may confuse agents communicating with a Multi-View TBox, we propose to prefer solutions with a minimal number of conflicts. We just learned that even for a simple Multi-View TBox like in Example 16, the choice might be nondeterministic when we only have TBox axioms at hand. In real-life applications, data exists in the form of instances.

In the presence of data, choosing the appropriate solution depends on the data. Instance-conflicts hence occur when the sub-concept of the axioms involved in a conflict are asserted instances. We hence need a qualitative measure that takes into account the instance-conflicts as well as the amount of information that is lost when certain entailments are not valid anymore.

6.2.2 Shannon Entropy

In computer science, information content is measured in terms of the *Shannon entropy* [Shannon, 1948]. More precisely, it is a measure of the average information content of a random variable we are missing when the value of the random variable is not known. The entropy $\mathcal{H}$ of a discrete random variable X with possible values $x_0, \ldots, x_N$ is defined as the expected value E of the information content I of X: $\mathcal{H} = E(I(X))$. The information content I is a random variable as well. If we know the probability mass function p of the random variable X, we may explicitly denote the entropy by:

$$\mathcal{H}(X) = -\sum_{n=0}^{N} p(x_n) \log_b p(x_n)$$

In case $p(x_n) = 0$, we have that $p(x_n) \log_b p(x_n) = 0$. The base b of the logarithm determines the unit in which the information content is measured. In computer science the information content is measured in in bits, so we use $b = 2$, as a result. We omit the subscript and simply write $\log p(x_n)$.

6.2.3 Entropy for Sets of Axioms

Existing approaches evaluating the information content of a TBox using entropy-based measures are not suitable for our case. These approaches want to evaluate a modularized ontology, i.e. they assess how well the modules are independent sub-units of the original ontology. Furthermore, they work on the axioms of the TBox rather than on the deductive closure. Both does not hold for our case.

Structural Entropy

Structural entropy measures like presented in Doran et al. [2009] define the probability mass function w.r.t. the RDF-graph representation of an OWL-ontology. While these measures can be computed without having to perform reasoning, they may fail in preferring solutions with less conflicts and, even more significant, with fewer invalidated entailments.

Example 17 (Entropy)**.** *Assume the simple TBox from Example 16. According to [Doran et al., 2009], the probability mass function for the entropy is the normalized count of the number of outgoing edges from a node. This is equal to all elements of both $\mathcal{DT}^0$ and $\mathcal{DT}^1$. As a result the solutions do not differ w.r.t. the language level entropy.*

The structural entropy measure fails to prefer solutions with fewer conflicts, because detecting the conflict relies upon at least one implicit entailment. The entailment is implicit, because we required to not have direct contradictions. As as result, any procedure that prefers solutions with fewer conflicts has to perform inference on potential solutions for $\Delta(\mathcal{T})$, i.e. on Multi-View TBoxes.

ABox Entropy

Different from the approach that defines entropy on concept level, we define the entropy on axiom level directly. In our case, the random variables are TBox axioms whose information content is determined by the number of ABox instances that satisfy the axiom. In our view, the entropy of a TBox is always to be seen in the context of a non-empty ABox.

An entropy measure for assessing solutions.

Before we can define the entropy of a TBox we have to first define how axioms can be treated as random variables. In the context of an ABox $\mathcal{A}$, we want to assign an axiom $B \sqsubseteq A$ a higher probability the more individuals from $\mathcal{A}$ satisfy it. We remember from Section 2.1 that axiom satisfiability, i.e. $\mathcal{T} \models B \sqsubseteq A$, can be rewritten as $\mathcal{T} \models \top \sqsubseteq \neg B \sqcup A$. Hence, we define the probability of an axiom by counting the number of individuals occurring in an ABox $\mathcal{A}$ that can be asserted to the complex concept $(\neg B \sqcup A)$.

For a TBox $\mathcal{T}$ let $\mathcal{C}$ be the set of all left-hand-sides and all right-hand-sides of all axioms of $\mathcal{T}$ as well as their complement. We introduce the auxiliary indicator function $\mathbb{I}_{\mathcal{A}} : \mathcal{C} \times \mathcal{A}^{\mathcal{I}} \rightarrow \{0, 1\}$ such that for $C \in \mathcal{C}$ and $x \in \mathcal{A}^{\mathcal{I}}$ it holds that $\mathbb{I}(C, x) = 1$, if $\mathcal{A} \models C(x)$ w.r.t. $\mathcal{T}$ and 0 else. The indicator function states whether a concept assertion is in the deductive closure of an ABox $\mathcal{A}$ or not. It helps us defining the probability mass function for TBox axioms:

Definition 23. *Let $\mathcal{T}$ be a TBox and $\mathcal{A}$ be an ABox. The probability mass function for an axiom $B \sqsubseteq A \in \mathcal{T}$ w.r.t. $\mathcal{A}$ is defined as*

$$p_{\mathcal{A}}(B \sqsubseteq A) = \frac{\sum_{x \in \mathcal{A}^{\mathcal{I}}} \mathbb{I}(\neg B \sqcup A; x)}{\sum_{D \sqsubseteq C \in \mathcal{T}} \sum_{y \in \mathcal{A}^{\mathcal{I}}} \mathbb{I}(\neg D \sqcup C; y)}$$

To determine this probability mass function, we have to compute for all axioms $B \sqsubseteq A \in \mathcal{T}$ the number of assertions to A and $\neg B$ that are logical consequences w.r.t. $\mathcal{T}$ and $\mathcal{A}$. We denote these numbers by n_A and $n_{\neg B}$ and observe that we can compute $\sum_{x \in (\mathcal{A})^{\mathcal{I}}} \mathbb{I}(\neg B \sqcup A; x) = \max\{n_A, n_{\neg B}\}$.

The entropy of a TBox $\mathcal{T}$ w.r.t. an ABox $\mathcal{A}$ is then defined as

$$\mathcal{H}_{\mathcal{A}}(\mathcal{T}) = - \sum_{B \sqsubseteq A \in (\mathcal{T})} p_{\mathcal{A}}(B \sqsubseteq A) \log p_{\mathcal{A}}(B \sqsubseteq A)$$

Our notion of entropy does not have to be restricted to TBoxes. Indeed we require that deterministic ABox reasoning is possible w.r.t. the set of axioms on which the entropy is defined. We can define the entropy for Multi-View TBoxes in the same way. This, in turn, allows us to assess Multi-View TBoxes $\mathcal{DT}$ by a qualitative measure that takes into account its information content w.r.t. an instantiation in the form of an ABox $\mathcal{A}$.

For the Multi-View TBoxes of Example 16 we obtain the following entropies for an ABox that asserts each concept a different individual:

$$\begin{array}{rcl} \mathcal{H}_{\mathcal{A}}(\mathcal{DT}^0) & = & 23.23 \\ \mathcal{H}_{\mathcal{A}}(\mathcal{DT}^1) & = & 25.48 \\ \mathcal{H}_{\mathcal{A}}(\mathcal{DT}^2) & = & 19.07 \\ \mathcal{H}_{\mathcal{A}}(\mathcal{DT}^3) & = & 25.48 \end{array}$$

The result with minimal entropy has a minimal number of conflicts. However, we should note that the other solution with minimal conflicts has a relatively high entropy value. The disjointness axioms $D \sqsubseteq \neg C$ and $D \sqsubseteq \neg B$ are logical consequences of $\mathcal{DT}^1$ but not of $\mathcal{DT}^2$. Similarly, the subclass-inclusion axioms $D \sqsubseteq B$ and $D \sqsubseteq A$ are logical consequences for $\mathcal{DT}^2$ but not for $\mathcal{DT}^1$.

The more disjoint axioms the more assertions to the complement of a concept are logical consequences of a TBox. Since negation is necessary (but not sufficient) for conflicts we, in general, assume that solutions with fewer disjoints are preferable. We further assume that the entropy increases when asserting individuals both a concept and its complement are valid logical consequences.

6.2.4 Complexity

The overall complexity of computing the entropy for a single solution requires checking satisfiability of assertions to the subconcept and the super-concept for each axiom in the TBox. This number is bounded by $2 \cdot |\mathcal{T}|$. In real-world ontologies, this number is expected to be significantly lower, because a concept, respective its complement, usually occur in more than one axiom either as sub-concept or as super-concept. Anyway, we have to perform a number of concept memberships that is linear in the number of axioms in the TBox.

6.2.5 Compromise Solutions

If we accept to potentially invalidate more entailments we can get rid of conflicts as defined in Definition 22. We therefore limit the non-deterministic choice of the Minimal-Splitting-Δ-Operator to Θ-axioms whereas we always choose all Γ-axioms.

If we change the choice accordingly, then the resulting Multi-View TBox contains no conflicts. Let $\mathcal{DT}$ be a consistent

Default Multi-View TBox and $\mathcal{C} = \{U \sqsubseteq D, U \sqsubseteq \neg D\}$ be a conflict w.r.t. $\mathcal{DT}$. We can assume without loss of generality that there exists a root JUC $J^k_{\mathcal{T},U\sqsubseteq\bot}$ for $U \sqsubseteq \bot$ w.r.t. $\mathcal{T}$,[2] such that $J^k_{\mathcal{T},U\sqsubseteq\bot} \models \mathcal{C}$. Since *all* Γ-axioms for $J^k_{\mathcal{T},U\sqsubseteq\bot}$ are chosen for a partition $\mathcal{U}_n$ and at least one corresponding Θ-axiom is part of a partition $\mathcal{U}_m$ with $m < n$, we invalidate $J^k_{\mathcal{T},U\sqsubseteq\bot}$. As a consequence, we invalidate a justification for $U \sqsubseteq D$ or $U \sqsubseteq \neg D$ w.r.t. $\mathcal{DT}$.

Invalidation vs conflicts

There may indeed exist more than one justification that corresponds to the conflict. Invalidating these by the Unsat-Splitting we do so for all justifications for $U \sqsubseteq D$ or $U \sqsubseteq \neg D$ (or both) w.r.t. the Views of $\mathcal{DT}$. Hence, at most one of the axioms of a potential conflict can be a logical consequence of $\mathcal{DT}$. As a result, $\mathcal{DT}$ cannot contain any conflict as defined by Definition 22. The drawback is that we potentially invalidate more entailments.

[2] If this was not the case, then there would exist a root JUC for a super-concept of U.

7

Experimental Results

In order to empirically evaluate the Splitting-Δ-Operator we performed tests on several publicly available ontologies. One of the performance measures for the evaluations is the number of inferences that are lost w.r.t. the original TBox. In Section 7.2 we show how to approximate the deductive closure of a TBox that has unsatisfiable concepts.

Plan of This Chapter

The first evaluation in Section 7.3 aims towards a principal assessment of the two Splitting-Δ-Operators compared to the Diagnosis-Δ-Operators w.r.t. the number of inferences that we will loose in the best and in the worst case. The second evaluation, subject to Section 7.4, was performed according to the optimization strategy presented in Section 6.2.

7.1 Specifications

We will now provide information about the ontologies used for the experiments, as well as about the software that we used and the software that we implemented. Finally, we give information about the hardware on which we ran the experiments.

Ontologies

For the experiments, we used a set of ontologies that have been and still are widely used in the literature for Ontology Repair [Kalyanpur et al., 2005; Horridge et al., 2008; Shchekotykhin and Friedrich, 2010].[1] Some Statistics about these ontologies can be found in Table 7.1. Statistics about the number of unsatisfiable concepts and justifications can be found in Table 7.2.

Software

For computing justifications we used the black-box approach described in Kalyanpur et al. [2005]. All experiments were performed using the Pellet OWL 2 reasoner [Sirin et al., 2007], version 2.2.0, and the Manchester OWL API [Horridge and Bechhofer, 2008], version 3.0.

We implemented the All-Defaults-Splitting-Δ-Operator as well as the Minimal-Defaults-Splitting-Δ-Operator and full Ontology Repair. For computing all possible solutions to the latter two approaches we used a stack based approach that performs a depth-first search.

The software is available at WSL [2] and licensed under the GNU Public License (GPL) version 3.0.[3]

[1] http://www.mindswap.org/ontologies/debugging/

[2] http://www.wsl.ch/info/mitarbeitende/scharren/owl-defaults/

[3] http://www.gnu.org/licenses/gpl-3.0.txt

Ontology	**Chemical**	**Economy**	**MiniTambis**	**SWEET**	**Terrorism**	**Transport**	**Travel**
Expr.	$\mathcal{ALCHF}(\mathcal{D})$	$\mathcal{ALCH}(\mathcal{D})$	$\mathcal{ALCN}$	$\mathcal{ALCHOF}(\mathcal{D})$	$\mathcal{ALCHI}(\mathcal{D})$	$\mathcal{ALCH}(\mathcal{D})$	$\mathcal{SOIN}(\mathcal{D})$
Profile	OWL 2 DL	OWL 2	OWL2	—	OWL 2	OWL 2 DL	OWL 2 DL
RBox	4	3	0	10	52	5	1
TBox	110	577	173	2216	449	926	73
ABox	0	1045	0	353	424	226	20
GCI	0	0	0	1	0	0	0
Concepts	48	339	183	1537	100	445	35
Obj. Prop.		45	44	102	132	89	6
Data Prop.		8	0	19	91	4	4
Ann. Prop.		2	0	2	2	2	2
Indiv.	0	482	0	150	86	183	14

Table 7.1: Ontologies used for the experiments. For each ontology, we list its expressive power and—if matching—the corresponding OWL 2 profile. We also provide the number of axioms and assertions in the RBox, in the TBox and in the ABox. In addition to that we state how many axioms of these are GCIs. Besides the axiomatic information, we show the number of atomic concepts, the number of object and data properties, and the number of named individuals.

Ontology	Chemical	Economy	MiniTambis	SWEET
# JUC	282	67	46	1
max size	11	6	13	13
med size	8	4	8	13
min size	5	3	2	13
# root JUC	7	48	6	1
	(2%)	(70%)	(13%)	(100%)
max size	6	5	6	13
med size	5	3	5	13
min size	5	3	2	13

Ontology	Terrorism	Transport	Travel
# JUC	24	127	1
max size	6	9	3
med size	4	5	3
min size	3	3	3
# root JUC	6	38	1
	(25%)	(30%)	(100%)
max size	4	9	3
med size	3	4	3
min size	3	3	3

Table 7.2: Results for the justifications and the root justifications for the unsatisfiable concepts of the ontologies of Table 7.1.

Hardware

All experiments were performed on a machine with eight Intel®Xeon®cores X5570 with 2.93GHz with 80GB RAM. Since we did not optimize for runtime, these numbers are solely presented for the sake of completeness.

7.2 Finite Deductive Closure

We learned in Section 2.1 that the deductive closure for a TBox is infinite. Since we are able to only compute a finite set of axioms, we define the finite deductive closure $(\mathcal{T})^+$ for a TBox $\mathcal{T}$

as the union of certain logical consequences for concepts in the signature of $\mathcal{T}$:

Added condition that D must also be in the signature.

Definition 24 (Finite Deductive Closure)**.** *Let $\mathcal{T}$ be a TBox. The* finite deductive closure *of $\mathcal{T}$, denoted by $\mathcal{T}^+$ is defined as follows:*

$$
\begin{aligned}
(\mathcal{T})^+ &= (SC_{\mathcal{T}})^+ \cup (EQ_{\mathcal{T}})^+ \cup (DC_{\mathcal{T}})^+ \\
(SC_{\mathcal{T}})^+ &= \bigcup_{D \in sig(\mathcal{T})} \{D \sqsubseteq C \mid \mathcal{T} \models D \sqsubseteq C \ \wedge \ C, D \in sig(\mathcal{T})\} \\
(EQ_{\mathcal{T}})^+ &= \bigcup_{D \in sig(\mathcal{T})} \{D \equiv C \mid \mathcal{T} \models D \equiv C \ \wedge \ C, D \in sig(\mathcal{T})\} \\
(DC_{\mathcal{T}})^+ &= \bigcup_{D \in sig(\mathcal{T})} \{D \sqsubseteq \neg C | \mathcal{T} \models D \sqsubseteq \neg C \wedge C, D \in sig(\mathcal{T})\}
\end{aligned}
$$

Interesting entailments

Computing the above logical consequences for concepts in the signature of $\mathcal{T}$ only causes $(\mathcal{T})^+$ to be finite. Since a TBox is a finite set of axioms, its signature must be finite, too. Let C be an arbitrary concept. Hence, the number of concepts from $sig(\mathcal{T})$ that do or do not subsume Cor are equivalent to C is finite. Hence $(SC_{\mathcal{T}})^+$, $(DC_{\mathcal{T}})^+$, and $(EQ_{\mathcal{T}})^+$ are finite, and so must be $(\mathcal{T})^+$. As a result, we have that $\mathcal{T} \subseteq (\mathcal{T})^+ \subset (\mathcal{T})^\star$.

The concept subsumption hierarchy for an unsatisfiable concept U w.r.t. a TBox $\mathcal{T}$ might not be determinable. According to the OWL 2 Direct Semantics [Horrocks et al., 2009], for each concept C in the signature of a TBox $\mathcal{T}$ it is implicitly assumed that $\mathcal{T} \models \bot \sqsubseteq C$. In case of an unsatisfiable concept U we have that $\mathcal{T} \models U \sqsubseteq \bot$, and hence $\mathcal{T} \models U \equiv \bot$. This, in turn, implies that $\mathcal{T} \models U \sqsubseteq C$ for any concept C in the signature of $\mathcal{T}$.

Although the above axioms are valid entailments for $\mathcal{T}$, we are rather interested in those parts of the concept hierarchy that are not inferred because of the contradiction. This issue is illustrated by Example 4 (cf Section 2.1). We admit that this position is arguable, but we think that the intention behind a concept hierarchy is not to consider entailments that are caused by a conflict.

To only have the favored inferences in the deductive closure of a TBox $\mathcal{T}$ that has unsatisfiable concepts, we redefine the deductive closure on basis of the Diagnosis-Δ-Operator. Let $\mathbb{D}$ be a diagnosis for $\mathcal{T}$. By definition, $\Delta(\mathcal{T}) = \mathcal{T} \setminus \mathbb{D}$ and $\mathcal{T} \setminus \mathbb{D} \not\models C \sqsubseteq \bot$ for any C in the signature of $\mathcal{T}$. Hence $\Delta(\mathcal{T})$ is a regular, coherent TBox, and $(\Delta(\mathcal{T}))^+$ can be computed in the usual way using classical inference. We compute all diagnoses for $\mathcal{T}$, compute the finite deductive closure for each corresponding $\Delta(\mathcal{T})$ and take the union of these finite deductive closures as the *preferred deductive closure* $(\mathcal{T})^\times$.

Definition 25 (Preferred Deductive Closure)**.** *Let* $\mathbb{D}_0, \ldots, \mathbb{D}_D$ *be all possible diagnoses for a TBox* $\mathcal{T}$*. Let* Δ^d *be the Diagnosis-*Δ*-Operators for which* $\Delta^d_{\mathcal{T}} = \mathcal{T} \setminus \mathbb{D}_d$ *for* $0 \leq d \leq D$*. The* preferred deductive closure *of* $\mathcal{T}$ *is defined as* $\mathcal{T}^\times = \bigcup_{d=0}^{D} (\Delta^d(\mathcal{T}))^+$*.*

Note that there can be exponentially many diagnoses in the number of axioms in the root JUCs for $\mathcal{T}$. Hence, computing $(\mathcal{T})^\times$ is possible only for small TBoxes. Table 7.3 shows the number of possible diagnoses for the ontologies used in the experiments.

7.3 Comparison of Finite Deductive Closures

In order to assess the quality of the Defaults-Splitting-Δ-Operators, we evaluated the All-Defaults-Splitting-Δ-Operator (Section 5.1.1), the Minimal-Defaults-Splitting-Δ-Operator (Section 5.1.2) and the Diagnosis-Δ-Operator (Section 3.3.1) w.r.t. ontologies from Table 7.1. The performance measure for this experiment was the number of entailments that we loose w.r.t. the approximated deductive closure when applying the corresponding Δ-Operators. While the solution for the All-Defaults-

Splitting-Δ-Operator is unique, we present only the best result for the Minimal-Defaults-Splitting-Δ-Operator.

Let $\mathcal{T}$ be a TBox and let $(\Delta^i(\mathcal{T}))^+$ be the deductive closure when applying the operator Δ^i. We measure the number of entailments that we loose when applying Δ^i to $\mathcal{T}$ by $\delta_i = |(\mathcal{T})^+ \setminus (\Delta^i(\mathcal{T}))^+|$. A Δ-Operator Δ^i is considered better than a Δ-Operator Δ^j, if $\delta_i > \delta_j$. In case $\delta_i = \delta_j$, they are considered to perform equally well.

Ontology Repair is the most commonly used technique for solving unsatisfiable concepts. We hence compare the results of both Splitting-Δ-Operators with the best Diagnosis-Δ-Operator.[4] Since we have to compute all Diagnosis-Δ-Operators for the approximation of the deductive closure of the TBox anyway, we get this result for free.

We computed the results for the All-Defaults-, the best Minimal-Defaults- and the best Diagnosis-Δ-Operator by δ_{all}, δ_{sgl} and δ_r, respectively.

Results

We compare the finite deductive closures in two ways. First we compute the size of the deductive closure for the best solution for both the Minimal-Defaults-Δ-Operator and for the Diagnosis-Δ-Operator. Sine for the All-Defaults-Δ-Operator there exists only one solution, we take this solution as the reference. Second, we compare the deductive closures of the optimal solutions for all approaches to see whether the invalidated inferences differ. The results are shown in Table 7.3.

When comparing the best diagnosis solution with the best mini-mal-defaults solution, the number of entailments invalidated is always lower using the latter approach, for all ontologies. The

[4] We should note that we performed a full repair, i.e. we removed whole axioms.

number of invalidated entailments varies more for the single diagnoses and not so much for the Minimal-Defaults approach.

There exists a large number of possible diagnosis solutions compared to only a few for the Minimal-Defaults approach.

We also compare the results of the different approaches w.r.t. the inferences in which their finite deductive closures differ. The Minimal-Defaults-Δ-Operator never invalidates any inference that the All-Defaults-Δ-Operator preserves.

The best Minimal-Defaults-Δ-Operator always performs better than the best solution of the Diagnosis-Δ-Operator.[5] For all solutions we found that the set of invalidated inferences and the set of preserved inferences differ always. The Diagnosis-Δ-Operator outperforms the All-Defaults-Δ-Operator in all cases but for the the TERRORISM ontology.[6]

In most of the cases the Minimal-Defaults approach as well as the Diagnosis Approach have a unique optimal solution regarding the number of inferences invalidated. This is, however, not true for the *Chemical* ontology for which there exist several solutions that are optimal in the aforementioned sense.[7]

7.4 Optimizing the Splitting-Δ-Operator

In order to determine optimal solutions for the Minimal-De-faults-Splitting-Δ-Operator, we performed an empirical evaluation for the entropy-based performance measure presented in Section 6.2. We used the ontologies listed in Table 7.1. For the ontologies present we could compute all possible solutions and did hence not apply stochastic search.

For each ontology, we computed the entropy in two ways. First, we defined an artificial ABox $\tilde{\mathcal{A}}$ such that each concept in the signature of the TBox was asserted a fresh individual. We did

[5] Row "Min-Def./ Diag."

[6] Row "All-Def./ Diag."

[7] Row "Min.-Def. solutions"

not require the individuals to be distinct. Second, for ontologies for which an ABox was provided, we also computed the entropy w.r.t. this real ABox.

Results

The results of these experiments are presented in Tables 7.4, and 7.5. Since the results for the experiments with the real ABoxes provided constant values for the entropy while stating zero conflicts, we omit displaying them in detail. We found out that indeed the assertions made in the ontologies did only concern axioms that were not involved in root JUC.

As we can see, the only ontologies for which the number of conflicts can be optimized w.r.t. the artificial ABox are the *Sweet* and the *Economy* ontology. For all other ontologies, the number of conflicts is the same w.r.t. all possible Default TBox solutions.

For the *Sweet* ontology lower entropy values correspond to a lower number of conflicts but to more invalidated inferences. There exist, however not a global optimal solution w.r.t. the minimal entropy nor w.r.t. the number of conflicts. While one of the solutions with minimal conflicts is amongst those with minimal entropy, the entropy for another solution with minimal conflicts does not correspond to a minimal entropy. The solution that minimizes both, the entropy as well as the number of conflicts (no. 1) also refers to a minimal number of inferences in the finite deductive closure. Yet, the other solution with minimal conflicts preserves more inferences, but is also not optimal.

The results indicate a correlation between entropy and number of inferences in the finite deductive closure: The lower the entropy the fewer the number of inferences that are preserved. The solutions with maximal entropy preserve most inferences but contain the maximal number of conflicts.

For the *Economy* ontology, however, the situation at hand is quite different. Minimizing the entropy corresponds to maximizing the number of conflicts. On the other hand, the minimal number of conflicts corresponds to entropy values in the median range, but not to maximal entropy values.

As was the case for the *Sweet* ontology, minimizing the entropy refers to minimizing the number of inferences preserved. The solutions with a maximal entropy correspond to the median number of conflicts, but not, as was the case before, to a maximal number of conflicts.

In both cases, minimizing the number of conflicts does not result in minimizing the number of inferences preserved. According to this measure, solutions with a minimal number of conflicts range around the median.

Ontology	Chemical	Economy	MiniTambis	SWEET
Expr.	$\mathcal{ALCHF}(\mathcal{D})$	$\mathcal{ALCH}(\mathcal{D})$	$\mathcal{ALCN}$	$\mathcal{ALCHOF}(\mathcal{D})$
TBox ax.	110	577	173	2′216
RBox ax.	4	3	0	10
root JUCs	4	x	9	1
Diag. solutions	1′822	x	1′296	13
max.	**907**	**x**	**499**	**13′697**
med.	592	x	364	9′889
min.	339	x	351	7′211
All-Def.				
max.	**770**	**18′822**	**400**	**13′675**
Compare				
All-Def./ Diag.				
pres.	88	x	88	1′578
lost	177	x	187	1′600
common	682	x	312	12′097
Min.-Def./ All.Def.				
pres.	177	x	185	144
lost	0	x	0	0
common	770	x	400	13′675
Min.-Def./ Diag.				
pres.	146	x	90	1′641
lost	58	x	4	1′519
common	801	x	495	12′178
Min.-Def. solutions	5	41	2	22
max.	**947**	**23′395**	**585**	**13′819**
med.	947	23′179	584	13′762
min.	944	22′506	582	13′751

Table 7.3: Comparison of Deductive Closures of all possible Diagnosis- and all possible Minimal-Defaults-Splitting-Δ-Operators applied to the ontologies from Table 7.1. Both approaches are compared w.r.t. the number of non-trivial inferences for $\Delta(\mathcal{T})$. For each approach, the number of possible solutions as well as the results for the best (max), the worst (min) and the median are presented. For entries that are marked with an ”x”, no results could be obtained, because the Java $^{\text{TM}}$Virtual Machine always ran out of memory.

Ontology	Terrorism	Transp.	Travel
Expr.	$\mathcal{ALCHI}(\mathcal{D})$	$\mathcal{ALCH}(\mathcal{D})$	$\mathcal{SOIN}(\mathcal{D})$
TBox ax.	449	926	73
RBox ax.	52	5	1
root JUCs	6	38	1
Diag. solutions	32′835	x	2
max.	**512**	**x**	**89**
med.	476	x	87
min.	435	x	84
All-Def.			
max.	**497**	**8′621**	**127**
Compare			
All-Def./ Diag.			
pres.	160	x	40
lost	175	x	2
common	337	x	87
Min.-Def./ All.Def.			
pres.	74	x	2
lost	0	x	0
common	497	x	127
Min.-Def./ Diag.			
pres.	185	x	42
lost	126	x	2
common	386	x	87
Min.-Def. solutions	8		1
max.	**571**	**x**	**129**
med.	537	x	129
min.	511	x	129

Table 7.3: Comparison of Deductive Closures of all possible Diagnosis- and all possible Minimal-Defaults-Splitting-Δ-Operators applied to the ontologies from Table 7.1. Both approaches are compared w.r.t. the number of non-trivial inferences for $\Delta(\mathcal{T})$. For each approach, the number of possible solutions as well as the results for the best (max), the worst (min) and the median are presented. For entries that are marked with an ”x”, no results could be obtained, because the Java $^{\text{TM}}$Virtual Machine always ran out of memory.

No.	FDC	$\mathcal{H}$	# Conflicts
1	947	18569	29
2	947	18569	29
3	947	18763	29
4	947	19084	29
5	944	19186	29
Median #Conflicts			29

(a) Chemical

No.	FDC	$\mathcal{H}$	# Conflicts
1	511	73473	14
2	511	73477	14
3	531	74267	14
4	517	74294	14
5	543	74432	14
6	553	74477	14
7	561	74524	14
8	571	74758	14
Median #Conflicts			14

(b) Terrorism

No.	FDC	$\mathcal{H}$	# Conflicts
1	582	5690	0
2	585	5702	0
Median #Conflicts			0

(c) MiniTambis

No.	FDC	$\mathcal{H}$	# Conflicts
1	129	2488	1
2	129	2488	1
Median #Conflicts			1

(d) Travel

No.	FDC	$\mathcal{H}$	# Confl.
1	13′751	1′891′029	1
2	13′752	1′891′029	2
3	13′753	1′891′029	3
4	13′756	1′891′068	5
5	13′757	1′891′068	6
6	13′757	1′891′068	6
7	13′757	1′891′068	6
8	13′760	1′891′102	7
9	13′760	1′891′340	1
10	13′761	1′891′386	2
11	13′764	1′891′386	5
12	13′763	1′891′386	4
13	13′762	1′891′386	3
14	13′764	1′891′386	5
15	13′764	1′891′386	5
16	13′775	1′891′580	6
17	13′783	1′891′731	8
18	13′795	1′891′953	9
19	13′808	1′892′053	8
20	13′817	1′892′240	9
21	13′816	1′892′268	10
22	13′819	1′892′275	10
Median #Conflicts			8

(e) Sweet

Table 7.4: Comparison of solutions for the Minimal-Defaults-Δ-Operator for the (a) *Chemical*, (b) *Terrorism*, (c) *MiniTambis*, (d) *Travel*, and (e) Sweet ontologies. The list is sorted w.r.t. the entropy $\mathcal{H}$ of the single solutions which is compared with the number of axioms in the finite deductive closure (FDC) and the number of conflicts for each solution

No.	FDC	$\mathcal{H}$	# Confl.
1	22′523	548′309	69
2	22′527	548′397	68
3	22′604	550′262	68
4	22′675	551′541	69
5	22′679	552′216	69
6	22′748	553′413	69
7	22′810	553′426	69
8	22′818	553′513	68
9	22′787	553′518	68
10	22′824	553′601	67
11	22′800	553′894	58
12	22′752	554′059	69
13	22′824	554′210	59
14	22′883	555′347	69
15	22′898	555′957	59
16	22′898	555′957	59
17	22′907	556′977	56
18	22′915	557′039	55
19	22′915	557′039	55
20	22′954	557′269	69
21	22′962	557′598	59
22	22′970	557′673	58
23	22′970	557′705	59
24	22′985	558′691	55
25	23′016	558′703	58
26	23′040	559′422	58
27	23′040	559′422	58
28	23′040	559′422	58
29	23′061	559′861	59
30	23′069	559′955	59
31	23′069	559′955	59
32	23′053	560′344	55
33	23′084	560′724	58
34	23′090	561′431	60
35	23′129	561′611	59
36	23′129	561′611	59
37	23′129	561′611	59
38	23′129	561′611	59
39	23′129	561′611	59
40	23′165	561′922	58
41	23′155	563′157	60
	Median #Conflicts		59

Table 7.5: Comparison of solutions for the Minimal-Defaults-Δ-Operator for the *Economy* ontology. The list is sorted w.r.t. the entropy $\mathcal{H}$ of the single solutions which is compared with the number of axioms in the finite deductive closure (FDC) and the number of conflicts for each solution

Part IV

Reflection

8

Related Work

The work related to this thesis can be grouped into the following categories:

1. Computing explanations and justifications.
2. Approaches to conflict solving.
3. Default Logics and its variants.
4. Ontology evaluation measures.

In the following we will give an overview over the most important works in these areas and give pointers for obtaining further information.

Plan of This Chapter

We start with an literature overview for computing explanations and justifications in Section 8.1 and continue with related work w.r.t. conflict solving in Section 8.2. In the subsequent Section 8.3 we provide details on the different approaches to Default Logic, in particular with extending DLs the handle default information. We finish the chapter with pointers to work done in the area of ontology evaluation in Section 8.4.

8.1 Computing Explanations and Justifications

In recent years, much progress has been made in the task to explain why a conclusion can be drawn from a DL knowledge base by solely using axioms from the knowledge base itself. Schlobach and Cornet [2003] came up with minimal unsatisfiable preserving sub-TBoxes (MUPS) which can explain the reason for unsatisfiability of concepts.

Justifications for Ontology Repair

Kalyanpur et al. [2005] introduced justifications as a form of minimal explanation for an arbitrary entailment. They presented a glassbox-approach that extends the tableaux calculus for $\mathcal{SHOIN}$ by a tracing mechanism to find a single justification for an entailment η. Further justifications for this entailment are computed by successively applying a blackbox-approach that systematically extends an empty TBox $\mathcal{T}_U$ with axioms from the original TBox $\mathcal{T}$ until $\mathcal{T}_U$ satisfies the entailment η. Afterwards, axioms are removed from $\mathcal{T}_U$ until the entailment is no longer inferred. In turn, some of the removed axioms are re-added to $\mathcal{T}_U$ until η is again inferred. The process of adding and removing axioms converges to a minimal TBox that explains η, i.e. a justification for η. It could be shown that computing all justifications for an entailment is feasible even when using the tableaux calculus for drawing logical consequences [Kalyanpur et al., 2005]. It was further shown that this task can be performed using the theory of minimal hitting sets [Reiter, 1987b]. Finding justifications is supported by the OWL API as well as many state-of-the-art DL reasoners.

Working on parts of axioms only

Recent approaches try to compute fine-grained [Lam et al., 2008] or laconic justifications [Horridge et al., 2008] to consider only the conflict causing sub-parts of an axiom. A structural transformation is performed on the original TBox $\mathcal{T}$ which rewrites axioms into a shorter, i.e. simpler form. The transformation preserves all inferences from the original TBox $\mathcal{T}$. Axioms from the transformed TBox $\mathcal{T}'$ refer to sub-parts of the original axioms [1]. Hence, removing axioms or inferences from $\mathcal{T}'$ results in potentially less invalidated inferences in the deductive closure of $\mathcal{T}'$.

Alternative approaches formulate the set of justifications as a pinpointing formula [Baader and Peñaloza, 2008]. Justifications are expressed as a monotone property on an axiomatized input. They are encoded by monotone Boolean formulas over a set of propositional variables, one for each axiom in the TBox. The pinpointing formula is then a monotone Boolean formula over these propositional variables such that a minimal valuation of this formula corresponds to the axioms in the root JUCs.

Justifications on lattices

Most available state-of-the-art OWL 2 reasoners justifications provide with justifications rather than the pinpointing formula. Therefore, the justifications based approach was used in the scope of this thesis.

8.2 Approaches to Conflict Solving

Solving conflicts in formal logics has been a controversial research subject ever-since antiquity. As we saw in Section 2.4, even the definition of a conflict depends on the very logics used. Consequently, the methods for solving conflicts for an ontology too depend on the underlying logical formalism.

8.2.1 Reasoning with Inconsistent Knowledge Bases

Huang et al proposed to perform reasoning on consistent subparts of the ontology [Huang et al., 2005]. Similar to choosing

[1] They refer to whole axioms of the original TBox, if these were already in their simple-most form

Reasoning on consistent sub-parts

the optimal partition solution by an ABox (Section 6.2), the subparts are selected according to a specific query. We meet all of the desired properties. However, besides this approach allows for a flexible reasoning process it is very expensive: for each query, a number of consistency checks has to be performed that is polynomial in the size of the ontology whereas the presented approach answers to queries in a constant number of satisfiability checks.

Yet another approach aims at completing DL knowledge bases [Sertkaya, 2008]. Similar to the case of Probabilistic Description Logics, the TBox is separated into a Universal TBox (called static there) and a part whose axioms can be refutable. Based on the refutable axioms, questions are generated in a certain way and presented to domain experts, i.e. human agents. By answering the questions, more axioms and assertions can be added to the knowledge base which becomes ”more complete”.

8.2.2 Ontology Repair

Removing axioms

In the area of resolving conflicts in ontologies, the main focus usually lies on resolving inconsistencies and hence changes mainly occur on instance level or rather restricted TBoxes. Haase and Völker avoided contradictions by computing a confidence on axioms and assertions [Haase and Völker, 2009] where unsatisfiability was restricted to disjoint- and subclass-axioms. Kalyanpur et al. [2005] investigated how justifications can be used for resolving unsatisfiable concepts. Their aim was to semi-automatically generate repair-plans to assist knowledge-engineers in resolving unsatisfiable concepts. Repair, in turn, is seen as either axiom removal or axiom rewriting. In either case original information is lost, which violated the conservation of explicit information (property P4 in Section 1). Indeed, the presented approach was developed with the intention to use justifications but not to remove axioms for assuring that no concept is inferred unsatisfiable. Therefore, ontology repair is considered as the baseline against which the presented approach is to be compared.

We should not forget to note that not always a full repair has to be performed. Approaches like fine-grained and/or laconic and precise justifications, which were mentioned in Section 8.1, allow for repair on parts of the axioms only.

8.2.3 Markov Logic

Markov Logic [Domingos, 2008] extend first-order logic formulas by specifying a weight for each formula. While reasoning, those formulas with the higher score have precedence over those with lower scores. It is still possible to declare certain formulas as pseudo-crisp by assigning them an infinite weight.

Relaxation of FOL

As such, Markov Logic provides a preference relation for free. Conflicts do not exist, but the score for certain entailments may be such that these may never become part of the deductive closure. In addition to that, the problem of estimating the scores remains ambiguous and depends on the task.

Nevertheless, Markov Logic, with an appropriate and efficient scoring procedure, could be a promising candidate for defining a Δ-Operator.

8.2.4 Repair Higher Order Logics

Repair can also be done using higher-order logics like in the Ontology Repair System [Bundy, 2007; Chan and Bundy, 2008]. It is assumed that some latent information is missing in the present axiomatic description of the world. This missing information, in turn, leads to contradictions. Since also higher-order repair assumes a monotone logic, the repair effectively makes changes to the structure of the ontology to resolve conflicts.

Besides that, Higher-order ontology repair, however, makes changes to the knowledge representation and cannot be applied to OWL ontologies in a straightforward way. It does, consequently, not match the required properties from Section 1.

8.2.5 Para-Consistent Logics

There have been made several proposals for a formal inconsistent-tolerant or *para-consistent logics*. A common feature of these is their ability to infer several types of conflicts, that would be contradictions in classical logics, not as a conflict. We will discuss those approaches that are most relevant to this work.

Jaśkowski's Discussive Logic

One of the first formal approaches to para-consistent logics is Jaśkowski's discussive logic or discussive propositional calculus [Jaśkowski, 1999]. The modal logics formula $\Diamond p$ is introduced which has the meaning "it is possible that p" holds. This possibility is defined on non-necessity $\Diamond p = \neg\Box\neg p$ where $\Box p$ means that "it is necessary that p holds" or, in other words "p occurs for all the possible events".

Beyond two-valued logic

A discussive system cannot be based on ordinary two-valued logic, because the elementary rule of modus ponens fails: If it is possible that $B \rightarrow A$ and is possible that B holds, then we cannot infer that it is possible that A holds. According to da Costa and Dubikajtis [1977], who extended Jaskowśki's propositional calculus to match higher-order logics, a discussive logic calculus

1. systematizes conceptions which contain contradictions,
2. handles theories containing contradictions caused by certain kinds of vagueness, and
3. treats directly some empirical disciplines whose main postulates are not consistent.

We consider the first property as resolving the unsatisfiable concept of a TBox. These are, in accordance with the second property, caused by the vagueness that occurs when ontologies evolve or are mapped onto each other. Inconsistencies are introduced when unsatisfiable concepts are instantiated. On the other hand, instances are ”real” data, and Collecting data is usually an empirical process. Meeting also the third property, our approach can be seen as a kind of discussive logic. Yet, the calculus used is supposed to be different from that of Jaśkowski's Discussive Logic.

Multi-Valued Logics

Multi-Valued logics may assign a logical formula more than two states. Based on Belnap's four-valued logic [Belnap Jr., 1977], for example, Ma et al. [2004] introduced a four-valued variant of $\mathcal{SHOIN}(\mathcal{D})$ which they call $\mathcal{SHOIN}(\mathcal{D})4$. Each formula is assigned one of the four truth values $true$, $false$, $\top$, $\bot$. They extend $\mathcal{SHOIN}(\mathcal{D})$ by three additional types of concept inclusion axioms. Two of these allow to model exceptions from strict concept inclusion.

Multi-valued logics change the knowledge representation.
Thereby, the preservation of expressive power (property P1 in Section 1) is violated which causes multivalued logics not to be considered suitable candidates for defining a Δ-Operator.

8.3 Default Logics

Lehmann vs Reiter

Default Logics were first introduced by Reiter [1980]. Because of issues with semi-normal defaults Lehmann [1995] provided a simpler perspective on Default Logics. It is based on conditional knowledge bases Lehmann and Magidor [1992] and introduced the concept of partitions for normal defaults. A similar approach was proposed independently at the same time by Benferhat et al. [1993]. An approach for extending DLs by default rules that aimed into the same direction as the Defaults interpreted by Lehmann was proposed by Padgham et al. [1993] and further extended by Lambrix et al. [1998].

Extending Lehmann's Default Logics

Lukasiewicz [2008] applied Lehmann's approach to Description Logics and extended it by a separate TBox that contains all the axioms which still model crisp and not default knowledge. He also enriched defaults by belief intervals resulting in a probabilistic variant of the DL $\mathcal{SHOIN}(\mathbf{D})$ which is referred to as $P - \mathcal{SHOIN}(\mathbf{D})$ or Probabilistic Description Logics. While in this paper we make use of the partition approach enriched by a TBox, we do not consider the possibility of assigning the axioms belief intervals—although this should, in principle, be possible.

Defaults for Description Logics

There have been made propositions of how to incorporate default knowledge in DL knowledge bases. The earliest approach was proposed by Baader and Hollunder [1995] and is based on Reiter's Default Logics. It defines a preference relation on prerequisites of defaults and hence deals with non-normal defaults (in contrast to Lehmann's approach which is based on normal defaults). Based on approach of Baader and Hollunder several proposals have been made how to extend DLs with default knowledge like Dao-Tran et al. [2009] and Navarro et al. [2007].

To the best of our knowledge, P-$\mathcal{SHOIN}(\mathbf{D})$ [Lukasiewicz, 2008] is currently the only formalism providing default entailment w.r.t. Lehmann's lexicographical entailment for OWL DL knowledge bases. An implementation is available [Klinov, 2008].

The idea of reasoning on different DL knowledge bases separately is also subject to so-called Distributed Description Logics (DDL) [Borgida and Serafini, 2002]. A Default extension to DDL was proposed by Ma and Wei [2004] that also refers to Baader's approach, and hence non-normal defaults.

8.4 Ontology Evaluation

Most approaches to ontology evaluation do not aim for reducing the number of conflicts for choosing from two similar ontologies. In fact there is a variety of approaches that define evaluation procedures for ontologies for different purposes:

Similarity measures

Ontology evaluation can be used to measure the similarity between ontologies [Maedche and Staab, 2002]. This is useful, for example, when one is missing information in the currently used ontology and wants to look for the missing bits in similar ontologies. Similarity could, in principle, be applied for assessing the quality of solutions to different Δ-Operators by comparing the original TBox $\mathcal{T}$ with $\Delta^i(\mathcal{T})$, a potential solution. However, how to reduce the number of conflicts is not quite obvious.

Aspect and task-based evaluation

Gangemi et al. [2006] introduced a general procedure for ontology evaluation regarding certain pre-defined aspects. In contrast to that, task-based ontology evaluation [Sabou et al., 2007] makes use of specialized and hence task-specific algorithms for assessing the quality of different ontologies for this special task. Yet, both approaches aim towards comparing different ontologies rather than to assess two different versions of the same ontology.

Entropy based measures

A first entropy-based approach for evaluating ontologies was defined by Calmet and Daemi [2004]. It was refined by Doran et al. [2009] and remains to the best of our knowledge the only approach that is based on an information-theoretic approach towards assessing the quality of different ontologies. For a more detailed overview over information-theoretic measures, the interested reader is referred to the book of Cover and Thomas [1991].

9

Discussion

We will now discuss the findings we made in the scope of this thesis. Looking back at our propositions we will study the pros and the cons that come along assuming they hold. We will see which goals could be achieved and provide possible solutions for those that could not. Furthermore, we will see that the experiments support two of our hypotheses while a third must be rejected. Again, we will discuss the problems and give conclusions for looking for alternatives.

Plan of this Chapter

We start the discussion with a reflecting the theoretical properties of the investigated approach for solving contradictions in Section 9.1. In addition, we discuss the findings of the experiments in Section 9.2. We close the discussion by pointing out the possibilities our approach enables for practical applications but also the limiting factors in Section 9.3.

9.1 Theoretical Aspects

We investigated several theoretical aspects to solving contradictions. We introduced a general Δ-Operator and defined several approaches in terms of this operator. We investigated how the different Δ-Operators meet the required properties, whether they are consistency and what their complexity is.

Required Properties

Requirements

We required the Δ-Operator to meet the following requirements:

Proposition 5 (Repetition)**.** *Any formalism for ontology repair has to fulfill the following properties:*

P1 Preservation of expressive power: The formalism for knowledge representation is not changed.

P2 Coherency: No concept is inferred unsatisfiable[1]*.*

P3 Autonomy: The procedure shall work automatically.

P4 Conservation of explicit information: The original explicitly stated information should be kept.

P5 Conservation of implicit information: As little inferred information as possible shall be lost.

To cover the non-determinism of the approaches we formulate this as an additional property:

P6 Determinism: The procedure shall work deterministically.

[1] For the current work we concentrate on resolving unsatisfiable concepts. The procedure may, in principle, be extended to resolve unsatisfiable roles and—to a certain extent—assertions.

Please note that we assume that these properties are desirable for a formalism that solves conflicts. It is, of course, arguable whether, for example, property P4 shall be preferred over property P5. For the use-case of where non-experts in knowledge engineering throw in all their knowledge into a knowledge base where also reasoning shall be possible, we are convinced that deleting the users' information is a potential threat to the users' motivation.

A summary of the investigated approaches according to fulfilling these properties is shown in Table 9.1.

How the approaches meet the requirements

All of these approaches fulfill the strict properties **P1-P3** whereas the related approaches fail in doing so. It is clear that these properties depend on the task. For manual ontology repair, for example, preserving original information might not be an important issue. Nevertheless these properties reflect the author's personal experience with people editing ontologies who are non-experts in knowledge engineering. We also learned that people are not good at expressing knowledge in a logical consistent way. While we did not perform any user experiments to show the handiness of our approach, we still believe that users, in particular non-experts in knowledge engineering, can benefit from this approach.

We also learned that for fulfilling the soft properties **P4-P6** there does not exist a solution that is optimal w.r.t all of them. Every solution turned out to be indeed a compromise. We should note that, although using our Defaults-Δ-Operators all axioms are preserved (**P4**), we *weaken* the original information by considering axioms separately during reasoning. If we, for example, want to minimize the number of invalidated inferences (**P5**), we need to make a non-deterministic choice (**P6**) which, in turn, requires optimization procedures.

Approach	**P1**	**P2**	**P3**	**P4**	**P5**	**P6**
Minimal Defaults	yes	yes	yes	yes	yes	no
All-Defaults	yes	yes	yes	yes	no	yes
Diagnosis	yes	yes	yes	no	yes	no
All-Repair	yes	yes	yes	no	no	yes
Null	yes	yes	yes	yes	no	no
Rewriting Axioms	yes	yes	no	no	yes	no
Multi-valued Logics	no	yes	yes	—	—	—
Markov Logic	yes	yes	yes	yes	—	—
Higher-Order Logics	no	yes	yes	no	—	—

Table 9.1: Comparison of approaches for defining a Δ-Operator w.r.t. to the desired properties defined in Section 1. The upper part of the table denotes the investigated approaches, the lower part denotes related approaches (cf Section 8.2). Properties **P1**-**P3** have to be strictly met whereas properties **P4**-**P5** are subject to optimization. Property **P6** states whether the Δ-operator is deterministic.

Consistency

Restrictions that can be weakened

We proved that, under certain requirements, the resulting Multi-View TBoxes are coherent. We require that contradictions are not stated explicitly. Some of these restrictions can be weakened. For example, we may resolve direct contradictions by rewriting axioms without having to invalidate the original axiom. Note that none of the investigated ontologies contained neither logical nor direct contradictions. Indeed there were always some inference steps involved for concluding contradictions from the JUCs.

Direct contradictions are obvious contradictions. Small-scale ontologies are usually manufactured by only a few persons. Due to their explicitness we assume that direct contradictions are detected easily. There are fewer chances that they survive in the ontology until the moment the ontology is published.

Consistency is achieved by considering certain axioms separately when drawing logical consequences. Default Logics as defined by Lehmann [1995] and Lukasiewicz [2008] work exactly like this and especially the latter method was designed to work for Description Logics. In accordance with that, we defined a data structure that can deal with separate TBoxes, i.e. the Multi-View TBoxes. We showed how the partition process of Lehmann's Default Logics provides a way how this separation can be achieved, and that this process leads to a consistent Multi-View TBox that meets the properties that a method for solving contradictions should have.

Furthermore, we introduced a splitting scheme that allows computing a consistent Multi-View TBox without the need for satisfiability checks. The splitting works on the JUCs. Together with a dependency graph that maintains the dependencies between the JUCs, solutions can be constructed by only performing set operations.

Splitting scheme saves satisfiability-checks

The splitting scheme allows to improve the process of finding partitions for the Multi-View TBox such that fewer entailments become invalidated. We showed that only a subset of the possible choices for the axioms of the root JUCs suffices to preserve consistency. Yet, this choice is non-deterministic and the possible solutions result in different Multi-View TBoxes. These can invalidate a different number of entailments and contain a different number of conflicts whereby this choice requires further optimization. This optimization is discussed along with the complexity of the problem below.

The improvement results in potentially invalidating fewer inferences, but the resulting Multi-View TBox can still contain conflicts. These conflicts were shown not to cause problems for drawing logical consequences. Still, the presence of conflicts is not desired, because they may confuse agents that work with the ontology.

Finding conflicts requires reasoning.

We showed that finding these conflicts requires reasoning on the Multi-View TBox. On the other hand, the conflicts correspond to the solved contradictions. We hence know what are the candidates we have to check and do not have to perform a blind search for conflicts, as a result.

We can possibly achieve absence of conflicts at the price of invalidating more entailments. We saw that putting all axioms from the Γ-sets (and not just a single axiom as in the optimal case) leads to a Multi-View-TBox that does not contain conflicts. Yet, it remains for future work to see whether this solution can still outperform full Ontology Repair in the number of invalidated entailments.

Complexity

The complexity of the problem of finding a single solution is dominated by the problem of finding JUCs. This, in turn, depends on the complexity of the satisfiability problem for the actual DL. In case of a very expressive DL like $\mathcal{SROIQ}$, the satisfiability problem is 2 NEXPTIME complete in the number of axioms in the TBox.

Computing JUCs dominates the complexity.

Once the JUCs are known the problem of computing a single solution requires a quadratic number of set operations in the number of axioms in the root JUCs. As we saw in the experiments, this number is relatively small compared to the number of axioms in the TBox.

Actually, the computation of all JUCs is not desired, because we only need the root JUCs for the procedure to work. We defined a basic algorithm for computing solutions iteratively. This algorithm might, on the one hand, benefit from not having to compute all JUCs. On the other hand, it might suffer from having to recompute solutions, because of dependencies between JUCs that are discovered on the fly. This solution was not implemented in the scope of this thesis and its further investigation is left for future work.

Finding optimal solutions

Finding optimal solutions can be of high complexity. If the axiom choice is non-deterministic, then the problem of finding an optimal solution is not straightforward. Optimal solutions may still contain conflicts, and detecting these and/or assessing the quality of a Multi-View TBox w.r.t. the number of conflicts requires reasoning.

If the number of possible solutions is large, then the detection and/or assessment of conflicts can become infeasible. We showed that, once the root JUCs are known the process of finding solutions can be defined in terms of a stochastic search. Changing single axioms in the choices is not too complicated. Because of that we gave a proposal how Simulated Annealing can be applied for finding optimal solutions. Since the ontologies that were available for the experiments were small enough to compute and assess all possible solutions, we did not implement this optimization strategy.

To sum up, finding single solutions is feasible. Finding optimal solutions may be subject to a (stochastic) search process.

Complex vs Atomic Unsatisfiable Concepts

Complex concepts require more effort.

As is done in Lukasiewicz' Probabilistic Description Logics [Lukasiewicz, 2008], we focus on solving unsatisfiable concepts in the extended signature of a TBox. We do so for the same reasons as is done in Probabilistic Description Logics: we are able to solve atomic as well as complex unsatisfiable concepts and we can more easy apply Lehmann's Lexicographical Entailment, if desired. The drawback is that we have to potentially perform more satisfiability checks, since we have to perform the satisfiability checks for the atomic concepts anyhow. On the other hand, we reduce the risk for asserting individuals to an unsatisfiable complex concept. While we do not provide a formal proof, it is presumable that the approach can be changed without too much effort to ″just″ ensure that no atomic concept is inferred unsatisfiable.

9.2 Empirical Evaluation

Before we start the discussion on the experimental part of this work, we repeat the two hypotheses from Section 1.1:

Hypothesis 1. *Using our approach fewer inferences are invalidated than in the case of axiom removal.*

Hypothesis 2. *Minimizing the number of conflicts does not result in maximizing the number of invalidated inferences.*

Hypothesis 3. *The entropy-based measure minimizes the number of conflicts and the number of invalidated inferences.*

Number of Solutions

We saw that the number of possible solutions to the Minimal-Defaults-Δ-Operator is relatively low w.r.t. the number of possible solutions to diagnosis. We were hence able to compute results for all ontologies that we investigated. This result is not quite unexpected, since the possible choices are quite limited. We learned in Section 5.1 that, in the worst case, we have at most $2 \cdot K \cdot \max_{k=0,\dots K} |\Theta^k|$ possible solutions, where K is the total number of root JUCs. In practice, it turns out that the actual number of possible solutions is lower than that boundary.

Optimization

Optimization techniques can be applied to finding solutions. While we were able to compute results for all possible solutions for the Minimal-Defaults-Δ-Operator, this might not be feasible for larger ontologies—particularly in the presence of large ABoxes. A stochastic search scheme like Simulated annealing still seems to best address the optimization problem. Mutating solutions in the manner of genetic algorithms requires to check whether a solution is possible. Changes for the partitions with a low index can have severe consequences for the subsequent partitions. It is hence easy to compute a new neighboring solution when we make a different choice. We consider it more straightforward than mutating solutions until we found one that is admissible.

Evaluation of Hypothesis 1

No definite statement possible

The experimental results support our claim that our approach invalidates fewer inferences than the classical approach of removing axioms for ontology repair. The sets of invalidated inferences differ for all combinations of possible Default TBoxes and axiom removals. None of these sets is a strict superset of the other. Hence, the overall performance using our approach is better than axiom removal, but there may exist cases where we may have invalidated more important inferences than axiom removal does. This can be overcome by making important implicit entailments explicit by adding these axioms to the original TBox (and, consequently, to $\mathcal{DT}$)—as long as they do not introduce cycles or direct contradictions.

Our approach is restricted to solving conflicts for unsatisfiable concepts. In an OWL 2 knowledge base, however, roles may become unsatisfiable and inconsistencies (i.e. assertions that cause the ABox not to be a model of the TBox) may occur. Since OWL 2, the treatment of roles has become similar to the treatment of concepts. Hence, the approach of splitting sets of axioms that support a role conflict in the TBox may also be applied to unsatisfiable roles. To address inconsistencies we may use abduction to add a new concept for assertions causing an inconsistency. Alternatively, we could treat instances like concepts and apply our approach directly.

Evaluation of Hypothesis 2

Evidence for accepting hypothesis

In the simple cases, the number of conflicts is the same for all possible solutions for the Minimal-Defaults-Δ-Operator. These can neither validate nor falsify our working hypothesis. For simple ontologies, we can only try to apply the compromise solution presented in Section 6.2.5, i.e. always choosing all Γ-axioms for all partitions. Yet, we have to accept that we potentially invalidate more inferences, but since minimizing the number of conflicts did not necessarily correspond to minimizing the number of inferences preserved, this seems a promising option. The compromise solution, however, was not implemented and its investigation is left for future work.

The only ontologies that are suitable for making a statement about Hypothesis 2 are the *Sweet* and the *Economy* ontology. For the first we saw that minimizing the number of conflicts bears the risk of minimizing the number of invalidated inferences. Yet there is no indication that this holds in the general case. On the contrary most of the best solutions w.r.t. a minimal number of conflicts are distributed around the median of the number of invalidated inferences. We hence decide to accept the hypothesis.

For ontologies that are significantly more complex than those investigated in the experiments, computing the conflicts might become infeasible w.r.t. the number of possible solutions. Since the entropy measure may not be suitable we would have to find other optimization strategies to assess the quality of a Multi-View-TBox.

Evaluation of Hypothesis 3

Evidence for rejecting hypothesis

The results for the entropy-based measure were ambivalent. In the simple cases, the number of conflicts is the same for all possible solutions for the Minimal-Defaults-Δ-Operator. Hence, the entropy cannot give any improvement. For complex ontologies we observed both cases: (1) the entropy-based measure could help minimizing the number of conflicts but (2) using the entropy-based measure also resulted in maximizing them. In any case, minimizing the entropy came along with minimizing the number of inferences preserved. Hence, we do not consider it useful in its present form for minimizing the number of conflicts nor the number of inferences preserved. As a result, we have to reject Hypothesis 3. Yet, we have to note that we worked on very small ABoxes and a very small number of conflicts.

Complexity

We only tested our approach on a few rather small knowledge bases. The complexity of the whole procedure we presented is dominated by the computation of all root JUCs for all unsatisfiable concepts. However, when we evolve a knowledge base, incremental reasoning [Parsia et al., 2005] may significantly reduce the computation time. If there was a way how to maintain the root JUCs for a TBox, our procedure, and ontology-repair in general, computing all root JUCs becomes feasible even for large Semantic Web knowledge bases.

9.3 Possibilities and Limits

No solution for ABox, but for TBox for non-experts

One possible application from which our approach can benefit was ontology evolution for people that are not experts in knowledge engineering. We allow for a coherent TBox while people can throw in all their information into the knowledge base. While we are not able to ensure (ABox) consistency in all cases, we excluded one important case therefore: We never entail any concept unsatisfiable, and it is hence safe to assert individuals to the concepts in the extended signature.

If, for example, the axiom $D \sqsubseteq \neg C$ is included in the knowledge base, it is still possible to generate an inconsistency by asserting an individual i to both C and D. We can, however solve such inconsistencies by adding the concept $D \sqcap C$ to the knowledge base explicitly.

Allow for intermediate solutions.

We can force the knowledge base not to entail any concept unsatisfiable by applying our approach. Yet, this should be only an intermediate solution and not a permanent one. We do not assume that all the contradictions we are able to solve (or better ignore) refer to exceptional knowledge—we use Default Logics just as a tool. It can be expected that the more contradictions are introduced and to be solved the more the information in the knowledge base gets weakened. It would be interesting to find out to what extend the knowledge base still provides meaningful answers when the number of contradictions increases.

In the same context we have to limit the number of contradictions for complexity reasons. If the number of contradictions gets too large the number of JUCs can grow exponentially. Even though our approach is not based on satisfiability checks (besides finding the JUCs which we have to find out for repair, anyhow), finding solutions can become an intractable problem.

Incremental solution may overcome complexity limitations.

The present approaches targets towards solving all conflicts in a single step. We could overcome the limitation on the number of contradictions by performing an incremental repair. Such an approach was semi-automatic and involves users to actually choose the next solving step from a number of alternatives.

10

Conclusion and Outlook

We presented an approach for automatic contradiction-free ontology evolution. We proposed a set of properties that a method for solving contradictions should meet. These properties were defined on the basis of reports in the literature but also on personal experience. In essence, we want to keep everything as much as possible of the original knowledge base (knowledge representation, ability to reason, explicit and implicit information), but want to be able to deal with contradictions.

Conclusions

Solving contradictions requires invalidating entailments. We gave a formal definition for a general operator whose purpose is to invalidate certain entailments such that contradictions are solved. This Δ-Operator allows to compare several approaches regarding the properties we defined.

Ontology Repair is the state-of-the-art technique for solving contradictions in DL knowledge bases. We showed how different variants of Ontology Repair can be formalized in terms of the

Δ-Operator and what impact these have on the properties we defined. The major drawback of Ontology Repair is that we have to delete explicitly stated information. This gave us the motivation for constructing a method that can deal with contradictions without the need to delete information that an agent has stated explicitly and the deletion of which is not desirable.

Based on Lehmann's Default Logics and Lukasiewicz Probabilistic Description Logics we constructed an approach that can handle contradictions automatically under certain requirements. We argued that these restrictions are not very special and hence do not limit the application of our approach. We showed that this approach produces a consistent knowledge base without the need to delete explicit information.

Limit invalidation

We have to invalidate certain implicit information, but can limit this to a minimum. We experimentally verified our claim that our approach performs better in terms of invalidated entailments than axiom removal on several ontologies.

The complexity of finding a solution is dominated by the problem of finding the causes for contradictions, i.e. the justifications for unsatisfiable concepts. Since it was shown in the literature that this problem is tractable for real-world ontologies that contain several thousands of axioms, we conclude that our approach is applicable. Furthermore, we provided the basic definition for an iterative version that might benefit from not having to compute all of the justifications for unsatisfiable concepts.

Optimal solutions

Optimal solutions require non-deterministic choices. Finding optimal solutions may hence increase the complexity of the problem, depending on the size of the knowledge base and the optimality criterion. Optimal solutions can still contain conflicts. These are—unlike in classical DL—not a problem for drawing logical consequences. Yet they may confuse agents working with the knowledge base. We saw that optimal solutions w.r.t. a minimal number of invalidated inferences may not correspond to a minimal number of conflicts. We concluded that we need a more sophisticated measure for assessing the quality of solutions. Such a method should minimize both the number of conflicts and the number of invalidated inferences.

We proposed an entropy-based measure that assesses the quality of a possible solution in the presence of an ABox. The rationale behind using entropy was that conflicts refer to higher values in the entropy. While working in theory, the empirical evaluation, however, showed that the entropy method may lead to both optimal as well as sub-optimal solutions w.r.t the number of conflicts. Regarding the number of invalidated inferences, the entropy measure showed to usually refer to sub-optimal results. We hence conclude that the entropy based measure in its presented form is not adequate to obtain optimal solutions w.r.t. minimizing the number of conflicts as well as the number of invalidated inferences.

Entropy measure did not minimize conflicts.

The experiments showed that minimizing the number of conflicts does not minimize the number of inferences invalidated. This makes us confident that minimal conflicts is a desirable property.

Outlook

The ontologies that we used are rather small and old. Nevertheless, they are still used in state-of-the-art literature on error handling in DL and OWL 2 ontologies. In order to know up to what extent our approach scales, we need larger ontologies that contain unsatisfiable concepts and complex (and hence interesting) JUCs. Published ontologies are usually coherent, i.e. they do not contain contradictions. We hence propose to launch an interest group that collects large-scale ontologies with unsatisfiable concepts.

The entropy-based measure was not able to minimize the number of conflicts. We therefore provided a re-formulation of our procedure that can minimizes the number of conflicts but risks increasing the number of invalidated inferences. This approach was not yet implemented and remains for future work.

Ontology mapping (i.e. aligning one ontology to another) is another task that has to deal with contradictions. Hence, we believe that our approach is a promising candidate for dealing with contradictions in ontology mapping.

General Conclusion

Non-experts may benefit most.

Ontology evolution for non-experts in knowledge engineering is the application that can benefit massively from applying our approach. On the one hand, we do not have to expose the complexity of performing repair to the users. On the other hand, we are able to keep the specified information—a factor that we consider important for the users' motivation. As sketched above, our approach provides a method for handling contradictions without the need to remove explicit information. Considering the amount of information (explicit and implicit axioms) that is invalidated, our approach can outperform full Ontology Repair while its limiting factors do not prevent its applicability. Indeed we believe that using the presented approach has the potential to enable automatic conflict-free ontology evolution in practice—a central need for long living ontologies and, hence, for the Semantic Web.

Part V

Appendix

A Glossary

	Abbreviations
DL	Description Logic
FOL	first-order logics
JUC	justification for unsatisfiable concept

	Description Logics
$\mathbb{DL}$	set of all Description Logics
$\mathcal{L}$	a certain Description Logic
$\leq_{\mathbb{DL}}$	partial order on Description Logics w.r.t. expressive power
$\mathbb{L}$	maximal expressive power of the ontology $\mathcal{O}$

	Concepts and Individuals *Concepts and Roles are denoted by uppercase Latin letters.* *Individuals are denoted by lowercase Latin letters.*
$\top$	the top concept
$\bot$	the bottom concept
A, B	atomic concepts
C, D	complex concepts
R	simple roles
S	complex roles
a, b, c, d	individuals, instances
$\neg C$	complex concept: negation/complement of C
$\forall R.C$	complex concept: value restriction for role R to concept C
$\exists R.\top$	complex concept: limited existential quantification for role R
$\exists R.C$	complex concept: full existential quantification for role R to concept C
$\leq nR.C$	complex concept: (qualified) cardinality restriction for role R (to concept C), also for $\geq$
$\{a, b, c\}$	complex concept: nominals, closed finite set of individuals
U	unsatisfiable concept for a TBox $\mathcal{T} \models U \sqsubseteq \bot$

	Satisfiability
$\Delta^{\mathcal{I}}$	(non-empty) interpretation domain
$\mathcal{I}$	interpretation, model
$a^{\mathcal{I}}$, $C^{\mathcal{I}}$, $R^{\mathcal{I}}$	interpretation of a, C, and R
$\models$	logical consequence relation

	Axioms and Assertions *Axioms and assertions are denoted by lower-case Greek letters.*
$D \sqsubseteq C$	subclass-inclusion axiom, D is subsumed by C or C subsumes D
$D \equiv C$	equivalence axiom, D is equivalent to C (and vice versa)
$S \sqsubseteq R$	role-inclusion axiom, S is subsumed by R or R subsumes S
$Prop(R)$	role assertion, R has property $Prop$
$C(d)$	assertion of individual d to concept C.
$R(c, d)$	assertion of individuals c and d to role R.
α, β	ABox assertions $C(d)$ or $R(c, d)$
τ, ρ	subclass inclusion axioms $C \sqsubseteq D$, equivalence axioms $C \equiv D$, role inclusion axioms $R \sqsubseteq S$, or role assertions like $Irr(R), Trans(R)$
η	inference, entailment

	Sets of Axioms and Assertions *Sets of axioms and assertions are denoted by calligraphic uppercase letters.*
$\mathcal{T}$	TBox, set of subclass inclusion axioms and concept equivalence axioms
$\mathcal{R}$	RBox, set of role inclusion axioms and role assertions
$\mathcal{A}$	ABox, set of individual assertions
$\mathcal{O}$	Ontology, pair of TBox and RBox $\mathcal{O} = \langle \mathcal{T}, \mathcal{R} \rangle$
$\mathcal{K}$	Knowledge base, pair of ontology and ABox $\mathcal{K} = \langle \mathcal{O}, \mathcal{A} \rangle$

	Deductive Closures $\mathcal{X}$ is one of $\mathcal{K}$, $\mathcal{O}$, $\mathcal{T}$, $\mathcal{R}$ $\mathcal{A}$, and $\Delta(\mathcal{T})$
$(\mathcal{X})^\star$	infinite deductive closure for $\mathcal{X}$
$(\mathcal{X})^+$	finite deductive closure for $\mathcal{X}$
$(\mathcal{X})^\times$	preferred deductive closure for $\mathcal{X}$

	Justifications *Justifications and parts thereof are sets of axioms and denoted similarly.*
$J^k_{\mathcal{T},\eta}$	k-th Justification for $\mathcal{T} \models \eta$
$\Gamma^k_{U\sqsubseteq\bot}$	splitting set for justification $J^k_{U\sqsubseteq\bot}$ containing all axioms with U on left-hand-side
$\Theta^k_{U\sqsubseteq\bot}$	splitting set for justification $J^k_{U\sqsubseteq\bot}$ containing all axioms without U on left-hand-side

	Multi-View TBoxes *Multi-View TBoxes and parts thereof are sets of axioms and denoted similarly.*
$\mathcal{DT}$	Multi-View-TBox
$\mathcal{T}_\Delta$	TBox that holds universal information for a Multi-View TBox
$\mathcal{U}_n$	n-th Aspect TBox of a Multi-View TBox
$\mathcal{T}_\Delta \cup \mathcal{U}_n$	n-th View of a Multi-View TBox
$\Delta(\mathcal{T})$	Δ-Operator applied to $\mathcal{T}$

	Lattices *Lattices and their operators.*
L	Lattice
$\oplus$	meet-operator
$\otimes$	join-operator
$L_{\oplus}$	sub-lattice induced by the meet-operator
$L_{\otimes}$	sub-lattice induced by the join-operator
p_n	propositional variable
P	a finite set of propositional variables $P = \{p_0. \ldots, p_N\}$
ψ, ϕ	(monotone) Boolean formulas over P
$\mathcal{V}$	valuation of P

	Choices
c_k, d_k	local choice function for the transformation of $\mathcal{DT}$ w.r.t. $J^k_{\mathcal{T},U\sqsubseteq\perp}$
$\bar{c}_k$	inverse local choice
$(c)_k^K$	global choice for $\mathcal{DT}$

B

Proofs

B.1 Proof of Lemma 2

Proof. For a TBox $\mathcal{T}$ which does not contain any logical contradiction, let $J_{\mathcal{T},U\sqsubseteq\bot}$ be an arbitrary JUC that contains less than two axioms. Since justifications cannot be empty, let $U \sqsubseteq C$ be the only axiom in $J_{\mathcal{T},U\sqsubseteq\bot}$. Assume the superclass C was unsatisfiable. Then there would have to exist $J_{\mathcal{T},C\sqsubseteq\bot}$ and $J_{\mathcal{T},U\sqsubseteq\bot} \supset J_{\mathcal{T},C\sqsubseteq\bot}$. Since JUCs cannot be empty, $J_{\mathcal{T},U\sqsubseteq\bot} \geq 2$. Consequently, we have to assume that C is satisfiable.

By definition, an JUC must infer the unsatisfiable concept unsatisfiable by solely using the justification's axioms: $J_{\mathcal{T},U\sqsubseteq\bot} \models U \sqsubseteq \bot$. Since C was satisfiable w.r.t. $\mathcal{T}$, this concept must—due to monotonicity—also be satisfiable for any subset of $\mathcal{T}$, i.e. $J_{\mathcal{T},U\sqsubseteq\bot} \not\models C \sqsubseteq \bot$. Since any interpretation domain is defined as a non-empty set, and there are no constraints on the interpretation of U, we will always be able to find a non-empty interpretation $U^{\mathcal{I}}$. As a result, U is satisfiable w.r.t. $J_{\mathcal{T},U\sqsubseteq\bot}$ which contradicts the definition of justifications. □

B.2 Proof of Lemma 3

Proof. Assume that none of the root JUCs for the unsatisfiable concepts of a TBox $\mathcal{T}$ contains a direct contradiction. Let $J^k_{U \sqsubseteq \bot}$ be an arbitrary root JUC for a concept U in the signature of $\mathcal{T}$.

Let U be a (possibly complex) concept. Assume that $\Gamma^k_{U \sqsubseteq \bot}$ is empty. In that case, none of the axioms in $J^k_{U \sqsubseteq \bot}$ has U on the left-hand-side. Since U must occur in some axiom of $J^k_{U \sqsubseteq \bot}$ (we could not make any statement about the satisfiability of U otherwise), it must occur on the right-hand-side of at least one axiom in $J^k_{U \sqsubseteq \bot}$. To make a conclusion about the satisfiability of U, the axiom must have the $\top$-concept (or a concept equivalent to the $\top$-concept) on the left-hand-side. This, in turn, requires that $\neg U$ occurs as a disjunct on the right-hand-side of one of the axioms in $J^k_{U \sqsubseteq \bot}$. We hence have a direct contradiction which violates our assumption.

Assume that $\Theta^k_{U \sqsubseteq \bot}$ is empty. Hence, all axioms from $J^k_{U \sqsubseteq \bot}$ have U on the left-hand-side. In that case $\Theta^k_{U \sqsubseteq \bot}$ either contains a direct contradiction or $\Theta^k_{U \sqsubseteq \bot}$ consists of more than two axioms. Assume that, without loss of generality, this root JUC consists of the three axioms $U \sqsubseteq C_1$, $U \sqsubseteq C_2$, and $U \sqsubseteq C_3$. To have $J^k_{U \sqsubseteq \bot}$ entail $U \sqsubseteq \bot$ we have that $J^k_{U \sqsubseteq \bot}$ entails $C_1 \sqcap C_2 \sqcap C_3 \sqsubseteq \bot$ for all three axioms, but not pairwise (we just excluded this case).
Considering the concept $C = (C_1 \sqcap C_2 \sqcap C_3)$ to be equal to the complex concept $\tilde{C} = (C_1 \sqcap C_2) \sqcap C_3$. We can hence rewrite the three axioms as the two axioms $U \sqsubseteq (C_1 \sqcap C_2)$ and $U \sqsubseteq C_3$ which is a direct contradiction.

□

B.3 Proof of Theorem 6

Theorem 6 (Repetition from Section 4.4.7)**.** *The Default TBox produced by Algorithm 1 is consistent, if*

$$\begin{array}{rcl}
\left| \quad chooseThetaAxiom(\mathcal{U}_n, \mathcal{T}_\Delta, \Theta^k_{U\,\sqsubseteq\,\perp})\right| & \geq & 1 \quad if \quad n = 0. \\
\left| chooseGammaAxiom(\mathcal{U}_n, \mathcal{T}_\Delta, \Gamma^k_{U\,\sqsubseteq\,\perp})\right| & \geq & 1 \quad if \quad n > 0. \\
updateIndex(n) & = & n + 1
\end{array}$$

We proof Theorem 6 by induction. Assume that all root JUCs $J^k_{\mathcal{T},U\sqsubseteq\perp}$ for the unsatisfiable concepts of a TBox $\mathcal{T}$ are acyclic and do not contain direct contradictions.

It should be noted that $\mathcal{T}_\Delta$ may grow dynamically, because in each processing step, and hence in each step of the induction, we potentially add further axioms to $\mathcal{T}_\Delta$. This requires to show for each step that the condition given by Default Logics, i.e. $B \sqsubseteq A \in \mathcal{U}_n \iff \mathcal{T} \setminus \bigcup_{m=0}^{n-1} \mathcal{U}_m \models A \sqcap B$ is still fulfilled for every of the already computed partitions $\mathcal{U}_0, \ldots, \mathcal{U}_{n-1}$. This becomes clearer, since the condition can be rewritten as $\mathcal{T}_\Delta \cup \bigcup_{m=n}^{N} \mathcal{U}_m \models A \sqcap B$ and has to hold for all $n = 0, \ldots, N$.

Induction Assumption

As induction assumption we define that either all axioms have been processed—in that case the algorithm terminates—or a new partition $\mathcal{U}_n$ can be constructed such that for each axiom $B \sqsubseteq A \in \mathcal{U}_n$ the condition $\mathcal{T} \setminus \mathcal{T}_\Delta \cup \bigcup_{m=0}^{n-1} \mathcal{U}_m \models A \sqcap B$ is fulfilled. In the following we assume there are still axioms left to process.

Induction Start

For the first processing step, all those axioms from $\mathcal{T}$ are added to $\mathcal{T}_\triangle$ that are not part of any root JUC, i.e. $\mathcal{T}_\triangle = \mathcal{T} \setminus \bigcup_{k=0}^{K} J^k_{U\sqsubseteq\bot}$. As we know from Lemma 3, for each root JUC $J^k_{U\sqsubseteq\bot}$ the corresponding splitting sets $\Theta^k_{U\sqsubseteq\bot}, \Gamma^k_{U\sqsubseteq\bot}$ are non-empty. By assumption, the $J^k_{U\sqsubseteq\bot}$ are acyclic. Hence, we can always find some Θ-set whose axioms are not contained in any Γ-set. In other words: There exists $\Theta^k_{U\sqsubseteq\bot}$ such that $\Theta^k_{U\sqsubseteq\bot} \cap \Gamma^l_{V\sqsubseteq\bot} = \emptyset$ for all l, V. For each such $\Theta^k_{U\sqsubseteq\bot}$ we choose at least one of its axioms for $\mathcal{U}_0$ and add the rest to $\mathcal{T}_\triangle$. Please note that only those axioms that are added to the first partition $\mathcal{U}_0$ are treated as Defaults.[1] Those axioms added to $\mathcal{T}_\triangle$, in contrast, have regular DL semantics.

After the first processing step, $\mathcal{T}_\triangle \cup \mathcal{U}_0$ does not infer any concept unsatisfiable, i.e. for each axiom $B \sqsubseteq A \in \mathcal{U}_0$ it holds that $\mathcal{T}_\triangle \cup \mathcal{U}_0 \models A \sqcap B$. Hence the induction assumption is shown to be true for $n = 0$.

Induction Step

According to the induction assumption, in the n-th processing step, a new partition $\mathcal{U}_n$ can be constructed such that for each axiom
$B \sqsubseteq A \in \mathcal{U}_n$ the condition $\mathcal{T} \setminus \mathcal{T}_\triangle \cup \bigcup_{m=0}^{n-1} \mathcal{U}_m \models A \sqcap B$ is fulfilled.

When choosing axioms from Γ-sets of root JUCs, we always choose at least one axiom for the current partition and add the rest to $\mathcal{T}_\triangle$—in any case, the corresponding Γ-set is empty afterwards. According to Algorithm 1, in each processing step, we

[1] Treating an axiom as Default is to be interpreted in the same way as Lehmann does in Lehmann [1995]. It just means that the Default axioms are put into different partitions and only the axioms within a single partition are considered together when drawing logical consequences. Treating axioms as Defaults does, however, not necessarily mean that they have Default semantics as defined by lex-minimal entailment.

only choose those axioms $B \sqsubseteq A$ from Θ-sets of root JUCs in case $B \sqsubseteq A$ is not member of any Γ-set of any JUC. To sum up, after each processing step any Γ-set of a root JUC is either empty or no axioms have been removed from it at all.

Emptiness of Γ-sets: In case all Γ-sets of the root JUCs are empty, so must be the corresponding Θ-sets. If this was not the case, there must exist an axiom $B \sqsubseteq A$ that is contained in some $\Theta^{k_0}_{U_0 \sqsubseteq \bot}$ but not in any $\Gamma^{k}_{U \sqsubseteq \bot}$. Since we already added all the axioms of that kind to either $\mathcal{U}_0$ or $\mathcal{T}_\Delta$ in the first step (induction start), all root unsat Θ-sets must be empty when all root unsat Γ-sets are empty. In that case the procedure terminates.

Assume now that not all Γ-sets of the root JUCs are empty. To be able to find axioms for the next partition $\mathcal{U}_{n+1}$ we must show that in step n for at least one $\Theta^{k_0}_{U_0 \sqsubseteq \bot}$ of a root JUC all its remaining axioms were added to $\mathcal{U}_n$, i.e. there exists U_0, k_0 such that $\Theta_{U_0 \sqsubseteq \bot} = \emptyset$ after step n.

At least one Θ-set that was resolved in step n: Since we performed step n, we resolved some root unsat Γ-sets in that step. According to Algorithm 1, for $n > 0$, each axiom in a Θ-set is contained in a Γ-set at the start of a processing step. By adding some Γ-axioms to $\mathcal{U}_n$ and $\mathcal{T}_\Delta$, we thereby add some Θ-axioms to $\mathcal{U}_n$ and $\mathcal{T}_\Delta$, as well. Let $\Theta^{k_i}_{U_j}$ be the Θ-sets for which we effectively added axioms to $\mathcal{U}_n$ and $\mathcal{T}_\Delta$. Since we remove these axioms from all splitting sets, we remove them from $\Theta^{k_i}_{U_j}$. Hence, after step n each $\Theta^{k_i}_{U_j}$ is either empty or still contains some axioms. We have to show that at least one $\Theta^{k_i}_{U_j}$ is empty.

Assume that none of the $\Theta^{k_i}_{U_j}$ was empty. Because we added all Θ-axioms that are at step n not in any Γ-set already to either $\mathcal{U}_m$ $(m = 0, \ldots, n)$ or $\mathcal{T}_\Delta$, all axioms in $\bigcup_{j,i} \Theta^{k_i}_{U_j}$ are contained in some Γ-set. These Γ-sets either belong to a root JUC or at least depend on root a JUC. Let us refer to these root JUCs by $J^{k_p}_{U_l \sqsubseteq \bot}$.

These, obviously have not yet been solved (otherwise, their Γ-sets had already been processed). As a consequence, their Θ-sets $\Theta^{k_p}_{U_l \sqsubseteq \bot}$ cannot be empty. As a consequence, the induction assumption that we started with above holds for the sets $\Theta^{k_i}_{U_j}$. We hence showed by induction: If none of the $\Theta^{k_i}_{U_j}$ is empty, then we can find root justifications $J^{k_p}_{U_l \sqsubseteq \bot}$ whose Θ-sets are non-empty.

By assumption the number of root JUCs is finite and acyclic. We found a contradiction, as a consequence. Hence, we proved that after step n there exists at least one $\Theta^{k_i}_{U_j}$ that is indeed empty. According to Lemma 3 the corresponding splitting sets $\Gamma^{k_i}_{U_j}$ are not empty. From $\Gamma^{k_i}_{U_j}$ we can construct partition $\mathcal{U}_{n+1}$ according to Algorithm 1.

Default Logics constraint on axioms in $\mathcal{U}_{n+1}$: Each axiom $B \sqsubseteq A \in \mathcal{U}_{n+1}$ was contained in some splitting set of a root JUC: Either $B \sqsubseteq A \in \Gamma^{k_0}_{U_0}$ or $B \sqsubseteq A \in \Theta^{k_1}_{U_1}$ such that $B \sqsubseteq A \notin \Gamma^{k_i}_{U_j}$.
In the first case, at least one axiom from $\Theta^{k_0}_{U_0}$ is contained in the previous partition $\mathcal{U}_n$. Hence, B is satisfiable w.r.t. $\mathcal{T} \setminus \bigcup^{n}_{m=0} \mathcal{U}_m$ whereas A was satisfiable before. As a result $\mathcal{T} \setminus \bigcup^{n}_{m=0} \mathcal{U}_m \models B \sqcap A$.
In the second case B was unsatisfiable in step n w.r.t. a JUC $J^{k_2}_{U_2 \sqsubseteq \bot}$ that depended on a root JUC $J^{k_3}_{U_3 \sqsubseteq \bot}$ which was solved in step $n + 1$[2]. Since $B \sqsubseteq A$ does not belong to a Γ-set anymore, the concept B is satisfiable w.r.t. $\mathcal{T} \setminus \bigcup^{n}_{m=0} \mathcal{U}_m$. The concept A was already satisfiable before, and $\mathcal{T} \setminus \bigcup^{n}_{m=0} \mathcal{U}_m \models A \sqcap B$, as a result.

[2] In step n the axiom $B \sqsubseteq A$ was still part of a Γ-set. Since it was not solved by solving a Γ-set of a root JUC in any step including $n + 1$, it must have been part of a derived JUC. Since it, by definition, does not belong to any Γ-set anymore, the corresponding derived JUC must have been solved in the $n + 1$-th step.

B.3.1 Example

Assume the TBox

$$\mathcal{T} = \left\{ \begin{array}{lll} B \sqsubseteq A, & C \sqsubseteq B, & C \sqsubseteq \neg H, \\ H \sqsubseteq A, & D \sqsubseteq C, & D \sqsubseteq \neg E, \\ E \sqsubseteq B & F \sqsubseteq D & F \sqsubseteq \neg C \\ G \sqsubseteq B & & \end{array} \right\}.$$

There exist the following root JUCs:

1. $J^0_{C \sqsubseteq \bot} = \{\underbrace{B \sqsubseteq A, H \sqsubseteq A}_{\Theta}, \underbrace{C \sqsubseteq B, C \sqsubseteq \neg H}_{\Gamma}\}$
2. $J^1_{D \sqsubseteq \bot} = \{\underbrace{C \sqsubseteq B, E \sqsubseteq B}_{\Theta}, \underbrace{D \sqsubseteq C, D \sqsubseteq \neg E}_{\Gamma}\}.$
3. $J^2_{F \sqsubseteq \bot} = \{\underbrace{D \sqsubseteq C}_{\Theta}, \underbrace{F \sqsubseteq \neg C, F \sqsubseteq D}_{\Gamma}\}.$

Induction start and first partition

For the induction start we initialize $\mathcal{T}_\Delta = \{G \sqsubseteq B\}$. For the first partition $\mathcal{U}_0$ we have to at least one axiom from $\{B \sqsubseteq A, H \sqsubseteq A\}$. Since $E \sqsubseteq B \in \Theta^1_{D \sqsubseteq \bot}$ but not in any Γ-set, we may also choose this axiom for the first partition. This possible choice causes the number of partitions to be not unique—if we choose $E \sqsubseteq B$ for the first partition, we may end up with three partitions, if we do not choose it for the first partition, we may end up with four partitions.

Induction step with ϑ-axioms

Assume that we do not choose $E \sqsubseteq B$ for the first partition. For the second partition $\mathcal{U}_1$, we may choose one or both axioms from $\{C \sqsubseteq B, C \sqsubseteq \neg H\}$. If we chose $C \sqsubseteq B$, we may put $C \sqsubseteq \neg H$ into $\mathcal{T}_\Delta$, because $C \sqsubseteq \neg H \notin \Theta^1_{D \sqsubseteq \bot}$. If we chose $C \sqsubseteq \neg H$, in contrast, we should not put $C \sqsubseteq B$ into $\mathcal{T}_\Delta$, because $C \sqsubseteq B$ is one of the axioms in $\Theta^1_{D\sqsubseteq\bot}$—putting $C \sqsubseteq B$ into $\mathcal{T}_\Delta$ we risk not to be able to resolve $\Theta^1_{D\sqsubseteq\bot}$. Assume we chose $C \sqsubseteq B$ for $\mathcal{U}_1$. Since $C \sqsubseteq B$ is not the only axiom in $\Theta^1_{D\sqsubseteq\bot}$ we have not yet resolved $\Theta^1_{D\sqsubseteq\bot}$ by removing $\mathcal{T}_\Delta \cup \mathcal{U}_0 \cup \mathcal{U}_1$ from the splitting sets. Indeed, there still exists some ϑ-axiom, because $\Theta^1_{D\sqsubseteq\bot} \setminus (\mathcal{T}_\Delta \cup \mathcal{U}_0 \cup \mathcal{U}_1) = \{E \sqsubseteq B\}$. We may add $E \sqsubseteq B$ either to $\mathcal{T}_\Delta$ or to $\mathcal{U}_1$ and remove it from the splitting sets.

Induction step without ϑ-axioms

By removing $E \sqsubseteq B$ from the splitting sets, we resolved $\Theta^1_{D\sqsubseteq\bot}$. For the next partition $\mathcal{U}_2$, we therefore can only choose from the corresponding Γ-set, i.e. $\Gamma^1_{D\sqsubseteq\bot} = \{D \sqsubseteq C, D \sqsubseteq \neg E\}$. Assume we chose $D \sqsubseteq C$ for $\mathcal{U}_2$ and $D \sqsubseteq \neg E$ for $\mathcal{T}_\Delta$. We remove both axioms from the splitting sets and can now indeed not find any ϑ-axioms: $\Theta^2_{F\sqsubseteq\bot} \setminus (\mathcal{T}_\Delta \cup \mathcal{U}_0 \cup \mathcal{U}_1) = \emptyset$.

Convergence

We resolved $\Theta^2_{F\sqsubseteq\bot}$. As a result, we may choose one or both axioms from $\Gamma^2_{E\sqsubseteq\bot}$ for $\mathcal{U}_3$ in the last step. Since there are no more axioms to process, the procedure stops.

Possible Partitions

The choice of axioms at each step determines a tree. The number of axiom within the partitions may vary as may vary the actual number of partitions. The following Table B.1 provides two possible solutions. Please note that there may exist further solutions.

$\mathcal{T}_\Delta$	$\mathcal{U}_0$	$\mathcal{U}_1$	$\mathcal{U}_2$	$\mathcal{U}_3$
$B \sqsubseteq A, C \sqsubseteq \neg H$ $D \sqsubseteq \neg E, F \sqsubseteq D,$	$H \sqsubseteq A, E \sqsubseteq B$	$C \sqsubseteq B$	$D \sqsubseteq C$	$F \sqsubseteq \neg C$
$B \sqsubseteq A, C \sqsubseteq B,$ $F \sqsubseteq D, D \sqsubseteq \neg E$	$H \sqsubseteq A, E \sqsubseteq B$	$C \sqsubseteq \neg H, D \sqsubseteq C$	$F \sqsubseteq \neg C$	

Table B.1: Possible partitions for the example in Section B.3.1.

B.4 Proof of Theorem 8

Theorem 8 (Repetition from Section 5.2)**.** *Let $\mathcal{T}$ be a TBox that contains unsatisfiable concepts. There exists a splitting-Δ-operator for two aspect TBoxes $\mathcal{U}_0, \mathcal{U}_1$ such that the resulting Multi-View-TBox is consistent.*

We proof Theorem 8 by induction. Assume that all root JUCs for the unsatisfiable concepts of a TBox $\mathcal{T}$ are acyclic and do not contain direct contradictions.

Induction Assumption

The induction assumption is that we can always find some set $\Theta^k_{U \sqsubseteq \bot}$ such that $\Theta^k_{U \sqsubseteq \bot} \cap \Gamma^{k'}_{U' \sqsubseteq \bot} = \emptyset$. We choose one of these axioms for $\mathcal{U}_0$ and add the rest to $\mathcal{T}_\Delta$. We remove $\mathcal{T}_\Delta \cup \mathcal{U}_0$ from the splitting sets.

Let $\Gamma^k_{U \sqsubseteq \bot}$ be the corresponding sets for which we just solved their counterpart $\Theta^k_{U \sqsubseteq \bot}$. Because the root JUCs are acyclic, one axiom γ^k_0 of each $\Gamma^k_{U \sqsubseteq \bot}$ is not contained in any $\Theta^{k'}_{U' \sqsubseteq \bot}$. We add all these γ^k_0 to the second TBox $\mathcal{U}_1$. If a $\Gamma^k_{U \sqsubseteq \bot}$ contains a second axiom γ^k_1 which is not contained in any $\Theta^{k'}_{U' \sqsubseteq \bot}$, we add these axioms γ^k_1 to $\mathcal{T}_\Delta$. Again, we remove $\mathcal{T}_\Delta \cup \mathcal{U}_1$ from the splitting sets and start again with the Θ-sets. Neither $\mathcal{T}_\Delta \cup \mathcal{U}_0$ nor $\mathcal{T}_\Delta \cup \mathcal{U}_1$ infers any of its concepts unsatisfiable.

Induction Start

In a first step, we add all axioms from $\mathcal{T}$ that are not contained in any root JUC to $\mathcal{T}_\Delta$. As we know from Lemma 3, the splitting sets of the root JUCs are non-empty. Because the $J^k_{U\sqsubseteq\bot}$ are acyclic, we can always find some Θ-set whose axioms are not contained in any Γ-set. In other words: There exists $\Theta^k_{U\sqsubseteq\bot}$ such that $\Theta^k_{U\sqsubseteq\bot} \cap \Gamma^{k'}_{U'\sqsubseteq\bot} = \emptyset$ for all k', U'. For each such $\Theta^k_{U\sqsubseteq\bot}$ we choose one of its axioms for $\mathcal{U}_0$ and add the rest to $\mathcal{T}_\Delta$. We remove $\mathcal{T}_\Delta$ from the splitting sets afterwards.

Neither axiom $B \sqsubseteq A$ in $\mathcal{T}_\Delta$ was part of any root JUC or contained in a Θ-set but not in a Γ-set. For each Θ-set that contributed axioms for $\mathcal{T}_\Delta$ and $\mathcal{U}_0$ one axiom of the whole root JUC $J^k_{U\sqsubseteq\bot}$ is not contained in $\mathcal{T}_\Delta \cup \mathcal{U}_0$. As such, $\mathcal{T}_\Delta \cup \mathcal{U}_0$ does not have unsatisfiable concepts. In this first step, $\mathcal{U}_1 = \emptyset$ which causes all concepts in $\mathcal{T}_\Delta \cup \mathcal{U}_1 = \mathcal{T}_\Delta$ also to be inferred satisfiable.

Induction Step

In the n-th step we added one or two axioms from some $\Gamma^k_{U\sqsubseteq\bot}$ to $\mathcal{U}_1$. After removing the added axioms from the splitting sets, either $\Gamma^k_{U\sqsubseteq\bot} = \emptyset$ or there exists $\Theta^{k'}_{U'\sqsubseteq\bot}$ such that we have $\Gamma^k_{U\sqsubseteq\bot} \cap \Theta^{k'}_{U'\sqsubseteq\bot} \neq \emptyset$. In case all $\Gamma^k_{U\sqsubseteq\bot} = \emptyset$, then also all $\Theta^k_{U\sqsubseteq\bot} = \emptyset$, by induction assumption. In particular: the procedure terminates.

Assume now there existed $\Theta^{k'}_{U'\sqsubseteq\bot}$ such that we have that $\Gamma^k_{U\sqsubseteq\bot} \cap \Theta^{k'}_{U'\sqsubseteq\bot} \neq \emptyset$. Since the root JUCs are acyclic, we know $\Theta^{k'}_{U'\sqsubseteq\bot} \cap \Gamma^{k''}_{U''\sqsubseteq\bot} = \emptyset$ for all $\Gamma^{k''}_{U''\sqsubseteq\bot} \neq \Gamma^k_{U\sqsubseteq\bot}$. We can now proceed as in the induction start and continue until all splitting sets are empty. Since the number of axioms is finite, the procedure terminates eventually.

According to Lemma 3 the splitting sets were non-empty in the beginning. At least one axiom from each Θ-set is contained in $\mathcal{U}_0$ whereas at least one axiom from each Γ-set is contained in $\mathcal{U}_1$. Hence, none of the corresponding root JUCs is completely contained neither in $\mathcal{T}_\Delta \cup \mathcal{U}_0$ nor in $\mathcal{T}_\Delta \cup \mathcal{U}_1$. As a result, neither $\mathcal{T}_\Delta \cup \mathcal{U}_0$ nor $\mathcal{T}_\Delta \cup \mathcal{U}_1$ infers any of its concepts unsatisfiable.

List of Figures

List of Tables

List of Algorithms

Bibliography

Alchourron, C. E., Gärdenfors, P., and Makinson, D. (1985). On the Logic of Theory Change: Partial Meet Contraction and Revision Functions. *The Journal of Symbolic Logic*, 50(2):510.

Baader, F. (1996). A formal definition for the expressive power of terminological knowledge representation languages. *Journal of Logic and Computation*, 6(1):33–54.

Baader, F. and Hollunder, B. (1993). How to prefer more specific defaults in terminological default logic. In Bajcsy [1993], pages 669–674.

Baader, F. and Hollunder, B. (1995). Embedding defaults into terminological knowledge representation formalisms. *Journal of Automated Reasoning*, 14:149–180.

Baader, F. and Nutt, W. (2007). *Basic Description Logics*, pages 43–95. Cambridge University Press, Cambridge, MA, USA, 2nd edition.

Baader, F. and Peñaloza, R. (2008). Axiom pinpointing in general tableaux. *Journal of Logic and Computation*, 20:5–34.

Baader, F. and Sattler, U. (2001). An overview of tableau algorithms for description logics. *Studia Logica*, 69:5–40.

Bajcsy, R., editor (1993). *IJCAI-93, Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence, Chambé,ry, France, August 28-September 3, 1993*. Morgan Kaufmann Publishers Inc.

Bauer-Messmer, B. and Grütter, R. (2007). Designing a bilingual eco-ontology for open and intuitive search. In Gómez, J. M., Sonnenschein, M., Müller, M., Welsch, H., and Rautenstrauch, C., editors, *Information Technologies in Environmental Engineering: ITEE 2007 - Third International ICSC Symposium*, Environmental Science and Engineering, pages 143–152, Heidelberg. Springer.

Bauer-Messmer, B., Grütter, R., and Scharrenbach, T. (2008). Improving an environmental ontology by incorporating user-input. In Moeller, A., Page, B., and Schreiber, M., editors, *EnviroInfo 2008: Environmental Informatics and Industrial Ecology, 22nd International Conference on Environmental Informatics*, Berichte aus der Umweltinformatik, pages 559–566, Aachen. Shaker.

Belnap Jr., N. D. (1977). A useful four-valued logic. In Jon Michael Dunn, G. E., editor, *Modern uses of multiple-valued logic : invited papers from the fifth International symposium on multiple-valued logic : Bloomington - India, May 13-16, 1975*, pages 8–37, Dordrecht, Holland. D. Reidel Publishing Company.

Benferhat, S., Cayrol, C., Dubois, D., Lang, J., and Prade, H. (1993). Inconsistency management and prioritized syntax-based entailment. In Bajcsy [1993], pages 640–647.

Borgida, A. and Serafini, L. (2002). Distributed description logics: Directed domain correspondences in federated information sources. In *Confederated International Conferences CoopIS, DOA, and ODBASE 2002, Irvine, California, USA, October 30 - November 1, 2002, Proceedings*, volume 2519 of *Lecture Notes in Computer Science (LNCS)*, pages 36–53. Springer Verlag.

Bundy, A. (2007). Where's my stuff? An ontology repair plan. In *Workshop on DISPROVING - Non-Theorems, Non-Validity, Non-Provability*, volume 4, pages 2–11. CADE Inc.

Calmet, J. and Daemi, A. (2004). From entropy to ontology. In *Cybernetics and Systems 2004 - AT2AI-4: From Agent Theory to Agent Implementation*.

Ceraso, J. and Provitera, A. (1971). Sources of error in syllogistic reasoning. *Cognitive Psychology*, 2:400–410.

Chan, M. and Bundy, A. (2008). Inconstancy: An ontology repair plan for adding hidden variables. In *Symposium on Automated Scientific Discovery*. AAAI Press.

Clark, K. L. (1987). Negation as failure. In Ginsberg [1987], pages 311–325.

Cover, T. M. and Thomas, J. A. (1991). *Elements of Information Theory*. Wiley-Interscience.

da Costa, N. and Dubikajtis, L. (1977). On Jaśkowski's discussive logic. In da Costa, N. and Chuaqui, R., editors, *Non-classical logics, model theory, and computability : proceedings of the Third Latin-American Symposium on Mathematical Logic, Campinas, Brazil, July 11-17, 1976*. North-Holland Publishing Company.

Dao-Tran, M., Eiter, T., and Krennwallner, T. (2009). Realizing default logic over description logic knowledge bases. In Sossai, C. and Chemello, G., editors, *Symbolic and Quantitative Approaches to Reasoning with Uncertainty, 10th European Conference, ECSQARU 2009, Verona, Italy, July 1-3, 2009, Proceedings*, volume 5590 of *LNCS/LNAI*, pages 602–613. Springer.

Domingos, P. (2008). *Markov Logic*, pages 92–117. Springer, Berlin.

Doran, P. S., Tamma, V., Payne, T. R., and Palmisano, I. (2009). An entropy inspired measure for evaluating ontology modularization. In *Proceedings of the Fifth International Conference on Knowledge Capture : Redondo Beach, California, USA, September 1-4, 2009*, pages 73–80. ACM Press.

Ferreiros, J. (2001). The road to modern logic-an interpretation. *The Bulletin of Symbolic Logic*, 7(4):pp. 441–484.

Frehner, M. and Brändli, M. (2006). Virtual database: Spatial analysis in a web-based data management system for distributed ecological data. *Environmental Modelling & Software*, 21:1544–1554.

Gangemi, A., Catenacci, C., Ciaramita, M., and Lehmann, J. (2006). Modelling ontology evaluation and validation. In Sure, Y. and Domingue, J., editors, *The Semantic Web: Research and Applications: 3rd European Semantic Web Conference, ESWC 2006, Budva, Montenegro, June 11-14, 2006, Proceedings*, volume 4011 of *LNCS*, pages 140–154, Heidelberg. Springer.

Gärdenfors, P. (2003). *Belief Revision*. Cambridge University Press.

Ginsberg, M. L. (1987). Readings in nonmonotonic reasoning.

Grätzer, G. (1998). *General lattice theory*. Birkhäuser, Basel.

Haarslev, V. and Möller, R. (1999). Race system description. In Lambrix, P., Borgida, A., Lenzerini, M., Möller, R., and Patel-Schneider, P. F., editors, *Proceedings of the 1999 International Workshop on Description Logics (DL'99), Linköping, Sweden, 1999*, volume 22 of *CEUR workshop proceedings*, pages 130–132. M. Jeusfeld c/o Redaktion Sun SITE, Informatik V, RWTH Aachen.

Haase, P. and Völker, J. (2009). Ontology learning and reasoning—Dealing with uncertainty and inconsistency. In

Costa, P. C. G. D., d'Amato, C., Fanizzi, N., and und Kathryn B. Laskey, editors, *Uncertainty Reasoning for the Semantic Web 1: ISWC International Workshop, URSW 2005-2007, Revised Selected and Invited Papers: Pt. 1*, volume 5327 of *LNCS*, pages 45–55, Heidelberg. Springer.

Hegel, G. W. F. (1817). *Enzyklopädie der philosophischen Wissenschaften im Grundrisse (1817)*. in German.

Hegel, G. W. F. (1991). *The Encyclopedia Logic: Part 1 of the Encyclopaedia of Philosophical Sciences With the Zusatze*. Hackett Publishing Company. reprint of Hegel [1817].

Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P. F., and Rudolph, S. (2009). OWL 2 Web Ontology Language Primer. Technical report, The World Wide Web Consortium (W3C). http://www.w3.org/TR/2009/REC-owl2-primer-20091027/.

Horridge, M. and Bechhofer, S. (2008). The OWL API: A Java API for working with OWL 2 ontologies. In Dolbear, C., Ruttenberg, A., and Sattler, U., editors, *Proceedings of the Fifth OWLED Workshop on OWL: Experiences and Directions, Collocated with the 7th International Semantic Web Conference (ISWC-2008), Karlsruhe, Germany, October 26-27, 2008*, volume 432 of *CEUR workshop proceedings*. M. Jeusfeld c/o Redaktion Sun SITE, Informatik V, RWTH Aachen.

Horridge, M., Parsia, B., and Sattler, U. (2008). Laconic and precise justifications in OWL. In Shet et al. [2008], pages 323–338.

Horrocks, I., Kutz, O., and Sattler, U. (2006). The even more irresistible $\mathcal{SROIQ}$. In Doherty, P., Mylopoulos, J., and Welty, C. A., editors, *Proceedings, Tenth International Conference on Principles of Knowledge Representation and Reasoning (KR2006), Lake District of the United Kingdom, June 2-5, 2006*, pages 57–67. AAAI Press.

Horrocks, I., Parsia, B., and Sattler, U. (2009). OWL 2 Web Ontology Language Direct Semantics. Technical report, The World Wide Web Consortium (W3C). http://www.w3.org/TR/2009/REC-owl2-direct-semantics-20091027/.

Horrocks, I. and Patel-Schneider, P. F. (1999). Optimizing description logic subsumption. *Journal of Logic and Computation*, 9(3):267–293.

Horrocks, I. and Sattler, U. (2004). Decidability of shiq with complex role inclusion axioms. *Artificial Intelligence*, 160:79–104.

Horrocks, I., Sattler, U., and Tobies, S. (1999). Practical reasoning for expressive Description Logics. In Ganzinger, H., McAllester, D., and Voronkov, A., editors, *Logic Programming and Automated Reasoning: 6th International Conference, LPAR'99, Tbilisi, Georgia, September 6-10, 1999, Proceedings*, LNAI, pages 161–180. Springer.

Huang, Z., van Harmelen, F., and ten Teije, A. (2005). Reasoning with inconsistent ontologies. In Kaelbling, L. P. and Saffiotti, A., editors, *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, July 30-August 5, 2005*, pages 254–259. Professional Book Center.

Jaśkowski, S. (1948). Rachunek zdań dla systemów dedukcyjnych sprzecznych. *Studia Societatis Scientiarum Torunensis*, Sectio A–I:57–77. in Polish.

Jaśkowski, S. (1999). A propositional calculus for contradictory deductive systems. *Logic and Logical Philosophy*, 7:35–56. English reprint of Jaśkowski [1948].

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

Kazakov, Y. (2008). $\mathcal{RIQ}$ and $\mathcal{SROIQ}$ are harder than $\mathcal{SHOIQ}$. In Brewka, G. and Lang, J., editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Eleventh International Conference (KR 2008), Sydney, Australia, September 16-19, 2008*, pages 274–284. AAAI Press.

Klinov, P. (2008). Pronto: A non-monotonic Probabilistic Description Logic reasoner. In Bechhofer, S., Hauswirth, M., Hoffmann, J., and Koubarakis, M., editors, *ESWC*, volume 5021 of *Lecture Notes in Computer Science*, pages 822–826. Springer.

Krötzsch, M., Vrandecic, D., Völkel, M., Haller, H., and Studer, R. (2007). Semantic wikipedia. *Journal of Web Semantics*, 5:251–261.

Lam, J. S. C., Sleeman, D., Pan, J. Z., and Vasconcelos, W. (2008). A fine-grained approach to resolving unsatisfiable ontologies. *Journal on Data Semantics*, pages 62–95.

Lambrix, P., Shahmehri, N., and Wahlloef, N. (1998). A default extension to description logics for use in an intelligent search engine. *Hawaii International Conference on System Sciences*, 5:28.

Lehmann, D. (1995). Another perspective on default reasoning. *Annals of Mathematics and Artificial Intelligence*, 15:61–82.

Lehmann, D. and Magidor, M. (1992). What does a conditional knowledge base entail? *Artificial Intelligence*, 55:1–60.

Levesque, H. J. and Brachman, R. J. (1987). Expressiveness and tractability in knowledge representation and reasoning. *Computational Intelligence*, 3:78–93.

Łukasiewicz, J. (1910). *O zasadzie sprzeczności u Arystotelesa (On the Principle of Contradiction in Aristotle)*. Kraków: Polska Akademia Umiejętności. in Polish.

Łukasiewicz, J. (1971). On the principle of contradiction in Aristotle. *Review of Metaphysics*, 24:485–509. translation of Łukasiewicz [1910] by Vernon Wedin.

Lukasiewicz, T. (2008). Expressive Probabilistic Description Logics. *Artificial Intelligence*, 172(6-7):852–883.

Lutz, C. and Miličić, M. (2007). A Tableau algorithm for Description Logics with concrete domains and general TBoxes. *Journal of Automated Reasoning*, 38(1-3):227–259.

Ma, Y., Lin, Z., and Lin, Z. (2004). Inferring with inconsistent OWL DL ontology: A multi-valued logic approach. In Lindner, W., Mesiti, M., Türker, C., Tzitzikas, Y., and Vakali, A., editors, *Current Trends in Database Technology - EDBT 2004 Workshops: EDBT 2004 Workshops PhD, DataX, PIM, P2P& DB, and ClustWeb, Heraklion, Crete, Greece, March 2004, Revised Selected Papers*, volume 4254 of *Lecture Notes in Computer Science (LNCS)*, pages 535–553. Springer.

Ma, Y. and Wei, J. (2004). A default extension to distributed description logics. In *IEEE/Wic/ACM International Conference on Intelligent Agent Technology, Proceedings, (Iat 2004), Beijing, China, September 20-24, 2004*, pages 38–44, Washington, DC, USA. IEEE Computer Society.

Maedche, A. and Staab, S. (2002). Measuring similarity between ontologies. In Gómez-Pérez, A. and Benjamins, V., editors, *Knowledge Engineering and Knowledge Management: Ontologies and the Semantic Web*, volume 2473 of *Lecture Notes in Computer Science (LNCS)*, pages 15–21. Springer Berlin / Heidelberg.

Meyer, T., Moodley, K., and Varzinczak, I. (2010). First steps in the computation of root justifications. In *2nd International Workshop on Automated Reasoning about Context and Ontology Evolution (ARCOE), Lisbon, Portugal, 16-17 August 2010*.

Motik, B. (2004). Kaon2 reasoner.

Motik, B., Grau, B. C., Horrocks, I., Wu, Z., Fokoue, A., and Lutz, C. (2009a). OWL 2 Web Ontology Language Profiles. Technical report, The World Wide Web Consortium (W3C). http://www.w3.org/TR/2009/REC-owl2-profiles-20091027/.

Motik, B., Shearer, R., Glimm, B., Stoilos, G., and Horrocks, I. (2009b). Hermit OWL reasoner.

Motik, B., Shearer, R., and Horrocks, I. (2009c). Hypertableau reasoning for description logics. *Journal of Artificial Intelligence Research*, 36(1):165–228.

Navarro, J. L., Sanchez, J. M., and Zurita, J. M. (2007). Default reasoning in Web Ontology Language. In *IADIS Multi Conference on Computer Science and Information Systems 2007, July 3-8 2007, Lisbon, Portugal, Proceedings*, pages 35–42. IADIS Press.

Nebel, B. (1988). Computational complexity of terminological reasoning in back. *Artificial Intelligence*, 34:371–383.

Padgham, L., Padgham, L., Zhang, T., and Zhang, T. (1993). A terminological logic with defaults: A definition and an application. In Bajcsy [1993], pages 662–668.

Parsia, B., Halaschek-wiener, C., and Sirin, E. (2005). Towards incremental reasoning through updates in OWL DL. In Carr, L., Roure, D. D., Iyengar, A., Goble, C. A., and Dahlin, M., editors, *Proceedings of the 15th international conference on World Wide Web, WWW 2006, Edinburgh, Scotland, UK, May 23-26, 2006*, New York, NY, USA. ACM Press.

Patel-Schneider, P. F. (1989). Undecidability of subsumption in nikl. *Artificial Intelligence*, 39:263–272.

Peñaloza, R. (2009). Reasoning with weighted ontologies. In Bernardo, Grau, C., Horrocks, I., Motik, B., and Sattler, U., editors, *Proceedings of the 2009 International Workshop on Description Logics (DL2009), Oxford UK, 2009*, volume 477 of *CEUR workshop proceedings*. M. Jeusfeld c/o Redaktion Sun SITE, Informatik V, RWTH Aachen.

Rector, A., Drummond, N., Horridge, M., Rogers, J., Knublauch, H., Stevens, R., HaiWang, and Wroe, C. (2004). Owl pizzas: Practical experience of teaching owl-dl: Common errors & common patterns. In Motta, E., Shadbolt, N., and Stutt, A., editors, *Engineering Knowledge in the Age of the Semantic Web: 14th International Conference, EKAW 2004, Whittlebury Hall, UK, October 5-8, 2004. Proceedings*, volume 3257 of *LNCS*, pages 63–81. Springer.

Reiter, R. (1980). A logic for default reasoning. *Artificial Intelligence*, 13(1-2):81–132.

Reiter, R. (1987a). On Closed World Data Bases. In Ginsberg [1987], pages 300–310.

Reiter, R. (1987b). A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95.

Russell, B. (1937). *The Principles of Mathematics*, page 102. George Allen & Unwin, London, 2nd edition.

Sabou, M., Gracia, J., Angeletou, S., D'Aquin, M., and Motta, E. (2007). Evaluating the semantic web: A task-based approach. In Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., and Cudré-Mauroux, P., editors, *The Semantic Web: 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, November 11-15, 2007*, volume 4825 of *LNCS*, pages 423–437. Springer Verlag, Heidelberg.

Scharrenbach, T. (2008). End-user assisted ontology evolution in uncertain domains. In Shet et al. [2008], pages 920–925.

Scharrenbach, T. and Bernstein, A. (2009). On the evolution of ontologies using Probabilistic Description Logics. In d'Amato, C., Fanizzi, N., Grobelnik, M., Lawrynowicz, A., and Svátek, V., editors, *Proceedings of the First ESWC Workshop on Inductive Reasoning and Machine Learning on the Semantic Web, Heraklion, Greece, June 1, 2009*, volume 474 of *CEUR workshop proceedings*. M. Jeusfeld c/o Redaktion Sun SITE, Informatik V, RWTH Aachen.

Scharrenbach, T., d'Amato, C., Fanizzi, N., Grütter, R., Waldvogel, B., and Bernstein, A. (2010a). Default logics for plausible reasoning with controversial axioms. In Bobillo, F., Carvalho, R., da Costa, P. C. G., D'Amato, C., Fanizzi, N., Laskey, K. B., Laskey, K. J., Lukasiewicz, T., Martin, T., Nickles, M., and Pool, M., editors, *Proceedings of the 6th International Workshop on Uncertainty Reasoning for the Semantic Web (URSW 2010), Shanghai, China, November 7, 2010*, volume 654 of *CEUR workshop proceedings*, pages 105–108. M. Jeusfeld c/o Redaktion Sun SITE, Informatik V, RWTH Aachen.

Scharrenbach, T., d'Amato, C., Fanizzi, N., Grütter, R., Waldvogel, B., and Bernstein, A. (2010b). Unsupervised Conflict-Free Ontology Evolution Without Removing Axioms. In Flouris, G. and Qi, G., editors, *Proceedings of the Fourth International Workshop on Ontology Dynamics (IWOD-2010), Shangai, China, November 8, 2010*, volume 651 of *CEUR workshop proceedings*, pages 55–68. M. Jeusfeld c/o Redaktion Sun SITE, Informatik V, RWTH Aachen.

Scharrenbach, T., Grütter, R., Waldvogel, B., and Bernstein, A. (2010c). Structure Preserving TBox Repair using Defaults. In Haarslev, V., Toman, D., and Weddell, G., editors, *DL2010: The 23rd International Workshop on Description Logics, Waterloo, Ontario, Canada May 4–7, 2010, Proceedings*, volume 573 of *CEUR* *workshop proceedings*. M. Jeusfeld c/o Redaktion Sun SITE, Informatik V, RWTH Aachen.

Schlobach, S. and Cornet, R. (2003). Non-standard reasoning services for the debugging of description logic terminologies. In Gottlob, G. and Walsh, T., editors, *IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9-15, 2003*, pages 355–362. Ap Professional.

Schmidt-Schauß, M. (1989). Subsumption in kl-one is undecidable. In Brachman, R. J., Levesque, H. J., and Reiter, R., editors, *Proceedings of the first international conference on Principles of knowledge representation and reasoning, Toronto, Ontario, May 15-18, 1989*, pages 421–431, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Schmidt-Schauss, M. and Smolka, G. (1991). Attributive concept descriptions with complements. *Artificial Intelligence*, 48:1–26.

Sertkaya, B. (2008). Explaining user errors in knowledge base completion. In Baader, F., Lutz, C., and Motik, B., editors, *Proceedings of the 21st International Workshop on Description Logics (DL2008), Dresden, Germany, May 13-16, 2008*, volume 353 of *CEUR Workshop Proceedings*. M. Jeusfeld c/o Redaktion Sun SITE, Informatik V, RWTH Aachen.

Shannon, C. E. (1948). A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 623–656.

Shchekotykhin, K. and Friedrich, G. (2010). Query strategy for sequential ontology debugging. In Patel-Schneider, P. F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J. Z., Horrocks, I., and Glimm, B., editors, *The Semantic Web - ISWC 2010: 9th International Semantic Web Conference, ISWC 2010, Shanghai, China, November 7-11, 2010, Revised Selected Papers, Proceedings*, volume 6496 of *LNCS*, pages 696–712, Heidelberg. Springer.

Shet, A., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., and Thirunarayan, K., editors (2008). *The Semantic Web - ISWC 2008: 7th International Semantic Web Conference, ISWC 2008, Karlsruhe, Germany, October 26-30, 2008, Proceedings*, volume 5318 of *LNCS*, Heidelberg. Springer.

Sirin, E., Parsia, B., Grau, B., Kalyanpur, A., and Katz, Y. (2007). Pellet: A practical OWL-DL reasoner. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(2):51–53.

Tarski, A. (1936a). O pojęciu wynikania logicznego. *Przegląd Filozoficzny*, pages 58–68. (in Polish).

Tarski, A. (1936b). Über den Begriff der logischen Folgerung. In *Actes du Congrès International de Philosophie Scientifique, Sorbonne, Paris, 1935, Vol. VII, Logique*, volume 394 of *Actualités Scientifiques et Industrielles*, pages 1–11, Paris. Hermann. (in German).

Tarski, A. (1983a). *Logic, Semantics, Metamathematics*. Hackett Publishing Company, 2nd edition.

Tarski, A. (1983b). *On the concept of logical consequence*. In Tarski [1983a], 2nd edition. Reprint of Tarski [1983c].

Tarski, A. (1983c). *On the concept of logical consequence*. In Tarski [1983a], 2nd edition. English translation of Tarski [1936a].

Tarski, A. (2002). On the concept of following logically. *History and Philosophy of Logic*, 23:155–196. translation of Tarski [1936b,a] by M. Stroińska and D. Hitchcock.

Tsarkov, D. and Horrocks, I. (2005). The Fact++ OWL-DL reasoner.

Tsarkov, D., Horrocks, I., and Patel-Schneider, P. F. (2007). Optimizing terminological reasoning for expressive description logics. *Journal of Automated Reasoning*, 39:277–316.

Ullman, J. D. (1988). *Principles of Database and Knowledge-Base Systems, Volume I*. Computer Science Press.

W3C OWL Working Group (2009). OWL 2 Web Ontology Language Document Overview. Technical report, The World Wide Web Consortium (W3C). http://www.w3.org/TR/2009/REC-owl2-overview-20091027/.

Curriculum Vitae

Personal Information

Name:	Thomas Scharrenbach
Date of Birth:	March 29, 1979
Citizenship:	German

Education

09/2011	Doctor in Informatics *End-User Assisted Ontology Evolution in Uncertain Domains*
02/2008 - 09/2011	PhD Student University of Zurich (Switzerland)
09/2005	Diploma (MSc) Computer Science *Minimum Bayes Risk for String Classification*
04/2003 - 11/2005	Student of Computer Science RWTH Aachen Technical University Department of Computer Science
09/1999 - 03/2003	Student of Computer Science University of Bonn Institute for Informatics
07/1998	Abitur (High-School Diploma) Bertha-von-Suttner Gymnasium, Andernach
08/1989 - 07/1998	Gymnasium (High-School) Bertha-von-Suttner Gymnasium, Andernach

Academic Expertise

01/2010	Visiting Faculty Università degli studi di Bari Aldo Moro Department of Informatics

Working Expertise

since 02/2011	Research Assistant University of Zurich Department of Informatics
02/2008 - 01/2011	PhD Student Swiss Federal Research Institute WSL
09/2007 - 01/2008	Freelancer IT-Consultant Aachen (Germany)
01/2006 - 08/2007	Research Associate RWTH Aachen Technical University Department for Mineral Processing AMR
01/2001 - 10/2004	Student Worker Agentur s11, Cologne, Germany
07/1997 - 08/1998	Zivildienst (Alternative Service) Fachklinik Bad Tönisstein Hospital for treatment of alcohol and drug addicts.