UNIVERSITÄT KOBLENZ · LANDAU
Faculty 4: Computer Science

WeST People and Knowledge Networks
Institute for Web Science and Technologies

# How I Lost My OWL: Retracting Knowledge from $\mathcal{EL}$ Concepts

## Master's Thesis

in partial fulfillment of the requirements for
the degree of Master of Science (M.Sc.)
in Informatik

submitted by
Holger Johannes Heinz

First supervisor: Prof. Dr. Steffen Staab
Institute for Web Science and Technologies

Second supervisor: Dr. Claudia Schon
Institute for Web Science and Technologies

Koblenz, May 2018

## Statement

I hereby certify that this thesis has been composed by me and is based on my own work, that I did not use any further resources than specified – in particular no references unmentioned in the reference section – and that I did not submit this thesis to another examination before. The paper submission is identical to the submitted electronic version.

| | Yes | No |
|---|---|---|
| I agree to have this thesis published in the library. | ☐ | ☐ |
| I agree to have this thesis published on the Web. | ☐ | ☐ |
| The thesis text is available under a Creative Commons License (CC BY-SA 4.0). | ☐ | ☐ |
| The source code is available under a GNU General Public License (GPLv3). | ☐ | ☐ |
| The collected data is available under a Creative Commons License (CC BY-SA 4.0). | ☐ | ☐ |

.....

(Place, Date) (Signature)

## Zusammenfassung

Ontologien sind wichtige Werkzeuge zur Wissensrepräsentation und elementare Bausteine des Semantic Web. Sie sind jedoch nicht statisch und können sich über die Zeit verändern. Die Gründe hierfür sind vielfältig: Konzepte innerhalb einer Ontologie können fehlerhaft modelliert worden sein, die von der Ontologie repräsentierte Domäne kann sich verändern oder eine Ontologie kann wiederverwendet werden und muss an den neuen Kontext angepasst oder mit bestehenden Ontologien verbunden werden. Die Schwierigkeit dieses Prozesses hat zur Entstehung des Forschungsfeldes der *Ontology Change* [1] geführt. Das Entfernen von Wissen aus Ontologien ist ein wichtiger Aspekt dieses Änderungsprozesses, da selbst das Hinzufügen neuen Wissens zu einer Ontologie das Entfernen bestehenden Wissens notwendig machen kann, falls dieses mit den neuen Vorstellungen in Konflikt steht [1]. Dieses Entfernen muss jedoch wohldurchdacht sein, da das Ändern bestehender Konzepte leicht zu viel Wissen aus der Ontologie entfernen oder die semantische Bedeutung der Konzepte auf eine potenziell unerwartete Weise verändern kann [2]. In dieser Arbeit wird daher ein formaler Operator zum präzisen Entfernen von Wissen aus Konzepten vorgestellt. Dieser basiert auf der Beschreibungslogik $\mathcal{EL}$ und baut partiell auf den Postulaten für Belief Set und Belief Base Contraction [3, 4, 5] sowie der Arbeit von Suchanek et al. [6] auf. Hierfür wird zunächst ein Einstieg in das Thema Ontologien und die Ontologiesprache OWL 2 gegeben und das Problemfeld der Ontology Change wird erläutert. Es wird dann gezeigt, wie ein formaler Operator diesen Prozess unterstützen kann und weshalb die Beschreibungslogik $\mathcal{EL}$ einen guten Ausgangspunkt für die Entwicklung eines solchen Operators darstellt. Anschließend wird ein Einblick in das Feld der Beschreibungslogiken gegeben. Hierfür wird die Geschichte der Beschreibungslogik kurz umrissen, Anwendungsgebiete werden genannt und es werden Standardprobleme in dieser Logik erläutert. In diesem Zusammenhang wird die Beschreibungslogik $\mathcal{EL}$ formal eingeführt. In einem nächsten Schritt werden verwandte Arbeiten untersucht und es wird gezeigt, warum das Recovery- und Relevance-Postulat für das Entfernen von Wissen aus Konzepten nicht unmittelbar anwendbar ist. Die hier gewonnenen Erkenntnisse werden anschließend dazu genutzt, die Anforderungen an den Operator zu formalisieren. Diese basieren hauptsächlich auf den Postulaten für Belief Set und Belief Base Contraction. Zusätzlich werden weitere Eigenschaften formuliert, welche den Verlust des Recovery- bzw. Relevance-Postulates ausgleichen sollen. In einem nächsten Schritt wird der Operator definiert und es wird gezeigt, dass diese Definition das präzise Entfernen von Wissen aus $\mathcal{EL}$-Konzepten gestattet. Mittels formaler Beweise wird zudem gezeigt, dass diese Definition alle zuvor aufgestellten Anforderungen erfüllt. In einem weiteren Beispiel wird dargestellt, wie der Operator in Verbindung mit sogenannten *Laconic Justifications* [7] verwendet werden kann, um einen menschlichen Ontology-Editor durch das automatisierte Entfernen von unerwünschten Konsequenzen aus der Ontologie zu unterstützen. Aufbauend auf Algorithmen, welche aus der formalen Definition des Operators abgeleitet wurden, wird ein Plugin zum Entfernen von Wissen aus Ontologien für den Ontology-Editor

Protégé vorgestellt. Anschließend werden die bisherigen Erkenntnisse zusammengefasst und es wird ein Fazit gezogen. Die Arbeit schließt mit einem Ausblick über mögliche zukünftige Forschung.


## Abstract

Ontologies are valuable tools for knowledge representation and important building blocks of the Semantic Web. They are not static and can change over time. Changing an ontology can be necessary for various reasons: the domain that is represented by an ontology can change or an ontology is reused and must be adapted to the new context. In addition, modeling errors could have been introduced into the ontology which must be found and removed. The non-triviality of the change process has led to the emerge of *ontology change* as an own field of research [1]. The removal of knowledge from ontologies is an important aspect of this change process, because even the addition of new knowledge to an ontology potentially requires the removal of older, conflicting knowledge [1]. Such a removal must be performed in a thought-out way. A naïve change of concepts within the ontology can easily remove other, unrelated knowledge or alter the semantics of concepts in an unintended way [2]. For these reasons, this thesis introduces a formal operator for the fine-grained retraction of knowledge from $\mathcal{EL}$ concepts which is partially based on the postulates for belief set contraction and belief base contraction [3, 4, 5] and the work of Suchanek et al. [6]. For this, a short introduction to ontologies and OWL 2 is given and the problem of ontology change is explained. It is then argued why a formal operator can support this process and why the Description Logic $\mathcal{EL}$ provides a good starting point for the development of such an operator. After this, a general introduction to Description Logic is given. This includes its history, an overview of its applications and common reasoning tasks in this logic. Following this, the logic $\mathcal{EL}$ is defined. In a next step, related work is examined and it is shown why the recovery postulate and the relevance postulate cannot be naïvely employed in the development of an operator that removes knowledge from $\mathcal{EL}$ concepts. Following this, the requirements to the operator are formulated and properties are given which are mainly based on the postulates for belief set and belief base contraction. Additional properties are developed which make up for the non-applicability of the recovery and relevance postulates. After this, a formal definition of the operator is given and it is shown that the operator is applicable to the task of a fine-grained removal of knowledge from $\mathcal{EL}$ concepts. In a next step, it is proven that the operator fulfills all the previously defined properties. It is then demonstrated how the operator can be combined with *laconic justifications* [7] to assist a human ontology editor by automatically removing unwanted consequences from an ontology. Building on this, a plugin for the ontology editor Protégé is introduced that is based on algorithms that were derived from the formal definition of the operator. The content of this work is then summarized and a final conclusion is drawn. The thesis closes with an outlook into possible future work.

# Contents

# Part I.
# Introduction and Motivation

## 1. Introduction

Following Studer et al. [8], who build upon Gruber [9] and Borst [10], an ontology is „[...] a formal, explicit specification of a shared conceptualisation“. Ontologies are a crucial part of the semantic web [11]. They provide a frame of reference through which machines can agree on common semantics of concepts and their relations and even infer new knowledge from them. Projects like DBPedia[1] aim at giving formerly unstructured knowledge a well-defined, ontology based structure. Today, a broad range of ontologies exists that aim at capturing the knowledge of different domains ranging from medicine [12, 13] and biology [14, 15, 16] to e-commerce [17] and social networks [18]. Some of these ontologies are considerably large: *SNOMED CT* is an ontology of medical and clinical terms. Its property holder, the *International Health Terminology Standards Development Organisation (IHTSDO)*, claims that it is „[...] the most comprehensive, multilingual clinical healthcare terminology in the world“ [19]. At the time of this writing, the International Edition of SNOMED CT contained more than 300,000 classes [20]. *The Gene Ontology (GO)* [14, 21] on the other hand is an ontology that is a part of *The Open Biomedical Ontologies* [22]. It aims at unifying the knowledge about genes and gene products and provides support for the description of cellular components, molecular functions and biological processes. At the time of writing, it contained more than 40,000 biological concepts and more than 6 million annotations of gene and gene products [23, 24].

The *OWL 2 Web Ontology Language* [25] is a formal language for the description of ontologies that was specifically tailored for the needs of the semantic web. It is a recommendation of the World Wide Web Consortium (W3C) since 2009. OWL 2 provides two different semantics: the OWL 2 RDF-based Semantics and the OWL 2 Direct Semantics. In addition, there are three trimmed down versions of OWL 2 available [26]: OWL 2 EL, OWL 2 QL and OWL 2 RL. They are called *OWL profiles* and contain only a specific subset of the full language. The removal of elements from the language reduces the expressive power of the profiles, but it also reduces the complexity of certain reasoning tasks.[2] All OWL 2 profiles use Description Logics as their formal underpinning which are a family of logic languages for knowledge representation. One important Description Logic is the logic $\mathcal{EL}$ and its extension $\mathcal{EL}$++ [27, 28] which corresponds to the OWL profile OWL 2 EL. $\mathcal{EL}$ and its extensions behave rather well in reasoning tasks but are expressive enough to model the content of several large ontologies, such as the aforementioned biomedical ontologies SNOMED CT and GO [27]. Catalyzed by its practical relevance, $\mathcal{EL}$ has

[1] See http://www.dbpedia.org.

[2] See Section 3.3.4 for an overview of common reasoning tasks.

attracted some attention in the scientific community. This includes the extension of $\mathcal{EL}$ with additional constructors [27, 28, 29, 30], its combination with temporal logic [31, 32] and its usage in the field of ontology based data access (OBDA) [33, 34].

## 2. Motivation

Imagine a world where the vision [35] of the Semantic Web has to its full extend become a reality. Most of the knowledge that is currently available on the World Wide Web is not limited to human eyes anymore, but can also be processed and understood by machines. These machines, called *agents*, work autonomously and are able to answer complex questions and perform sophisticated tasks by connecting a variety of decentralized sources of knowledge. In this process, these agents can also infer new knowledge through logical reasoning and share it with all participants of this web, thereby extending the knowledge of humanity. The rise of the semantic web as it was envisioned could therefore lead to a second dawn of the Information Age with an impact on everyday life that is hard to imagine.

For this vision to become true, certain obstacles have to be overcome. Today, ontologies that are based on well-defined languages such as OWL are important tools for knowledge representation and a cornerstone of the semantic web [11]. One of the problems that must be faced is how the knowledge that is stored in these ontologies can be modified [1, 36]. The reasons that make such a change necessary are manifold. To give an example, the reuse of ontologies provides certain benefits[3] and is of practical relevance[4]. However, in the case of its reuse the ontology must be adapted to the new context and the new users' needs. In addition, if multiple ontologies are merged together, it can be necessary to change them in order to base their content on a common vocabulary [1]. To give another reason for the change of an ontology, it can happen that the domain that is reflected by the ontology changes. Scientific research for example could alter our understanding of the human body and result in new medical knowledge. This knowledge would ultimately make its way into medical ontologies such as SNOMED CT. Unfortunately, performing changes to an ontology is sometimes far from trivial [1, 2].

The change of knowledge representation systems has been an active area of research and some work has been done with focus on schema evolution [41, 42, 43].[5] Unfortunately, methods for handling changes in database schemas cannot be applied offhandedly to the field of ontology change [45, 46, 47]. The problem of *ontol-*

[3] These benefits include a possible reduction of the development costs and the facilitation of a consistent description of the new domain in terms of previously used concepts and properties. Moreover, the reuse of an ontology can improve its quality as it facilitates its proofreading [37].

[4] Ochs et al. [38] for instance investigated ontology reuse in the National Center for Biomedical Ontology (NCBO) BioPortal [39, 40]. More than the half of 355 analyzed ontologies reused classes or properties from other ontologies.

[5] Rahm and Bernstein [44] provide an online bibliography with focus on schema evolution at http://se-pubs.dbs.uni-leipzig.de/.

*ogy change* has therefore become an own field of research.[6] Ontology change incorporates different aspects of the change process, such as ontology versioning, debugging, integrating and merging [1]. In many cases, a change of the ontology can make it necessary to remove existing knowledge [1, 36, 48]. Even a simple addition of new knowledge can introduce an inconsistency into the ontology as it can conflict with old beliefs [1]. If this is the case, the old knowledge that causes the conflict must be found and removed. Given a more concept-centric view, this includes the removal of knowledge from concepts that are stored in the ontology. Carried out by humans, the change of an ontology is prone to errors. Cornet et al. [2] for instance studied cases in which modifications of SNOMED CT lead to complex modeling problems such as the introduction of nonsensical clinical concepts or the creation of seemingly equal concepts which yet differed in their underlying logical representation.[7]

Given the non-triviality of this problem, it would therefore be desirable to have a well-defined operator at hand that supports the removal of knowledge from concepts. Such an operator would offer a clear semantics and could be defined in terms of desirable properties. These properties could cover basic aspects of the removal.[8] In addition, they could also include more sophisticated properties that limit the removal of knowledge from concepts and let the operator work in a fine-grained way. Furthermore, such an operator could assist a human editor in removing unwanted consequences from the ontology. For this, a program that employs the operator could first compute a set of *laconic justifications*[9] for an unwanted entailment. These laconic justifications identify the parts of the concepts that are responsible for the presence of the consequence. The program could then apply the operator to the concepts in order to automatically remove the concepts' parts that have been identified to cause the entailment.

For these reasons, the following part of this work is concerned with the development of an operator that supports a fine-grained removal of knowledge from concepts. The Description Logic $\mathcal{EL}$ is chosen as a starting point. This is motivated by the following: First, reasoning in $\mathcal{EL}$ and some of its extensions is tractable. This stands in sharp contrast to other Description Logics and is further illustrated in Section 3.3.4. Second, $\mathcal{EL}$ has a practical relevance since $\mathcal{EL}^{++}$, which is an extension of $\mathcal{EL}$, provides the logical base for the OWL profile OWL 2 EL and large biomedical ontologies. This provides the opportunity for a future extension of the operator to

[6] Flouris et al. [1] define ontology change as „[...] the problem of deciding the modifications to perform upon an ontology in response to a certain need for change as well as the implementation of these modifications and the management of their effects in depending data, services, applications, agents or other elements".

[7] Such errors can also stem from rather simple misconceptions of the ontology language. Rector et al. [49] gives an overview of common errors that were observed during the teaching of OWL. These errors included the misuse of concept names to express information about the domain and a wrong understanding of quantified statements, such as the trivial satisfiability of *allValuesFrom*.

[8] Such properties could for example ensure a success of the operation in that it is made sure that knowledge that is retracted is not present in the result anymore.

[9] Laconic justifications can be understood as fine-grained explanations for the entailment of a consequence. See Section 4.2.1 for a more detailed description.

$\mathcal{EL}^{++}$ and its application to real-world problems.

The next chapter gives an introduction to Description Logic with a focus on the Description Logic $\mathcal{EL}$ and examines related work.

# Part II.
# Background

In this chapter an introduction to Description Logic is given. For this, the history of Description Logic is outlined and some application areas are shown. Then the Description Logic $\mathcal{EL}$ is introduced and some additional definitions and examples are given. Following this, related work is examined in order to judge its applicability to the task of a fine-grained removal of knowledge from $\mathcal{EL}$ concepts. This includes the AGM postulates for belief set contraction and the postulates for belief base contraction. In addition, work is analyzed that is concerned with the debugging of ontologies and work that aims at computing the difference between Description Logic concepts.

## 3. Description Logic

This section gives an overview of the history of Description Logic and introduces the logic $\mathcal{EL}$. After this, some additional definitions are given. These definitions include the TBox and the ABox which together make up the knowledge base of Description Logic systems. It is then shown how the *canonical extension* and the *complete expansion* of a TBox can be computed. This section closes with a short overview of common reasoning tasks in Description Logic and their complexity in $\mathcal{EL}$ and some of its extensions.

### 3.1. History

Description Logics are a family of logic-based, formal languages. Figure 1 gives an overview of the evolution of Description Logic systems and reasoners. All dates in this figure show the time of the first appearance of a system. The displayed Description Logic is always the most recent logic that corresponds to a system or is supported by a reasoner.[10]

Description Logics stem from semantic networks [50] and frames [51]. These early ancestors differed from logic-based formalisms in that they had no well-defined formal semantics [65, 66]. In the case of semantic networks, Woods [66] for example criticized the ambiguity of links in these networks. They could be used to describe different parts of a conceptual structure or to express a relationship between concepts. To illustrate his critique, Woods gave the two concepts *black* and *telephone* and defined a new concept from them. He then argued that the new concept could either denote a black telephone or express the assertion that all telephones are black.

[10] Note that not all systems support ABox reasoning or provided sound and complete reasoning algorithms.

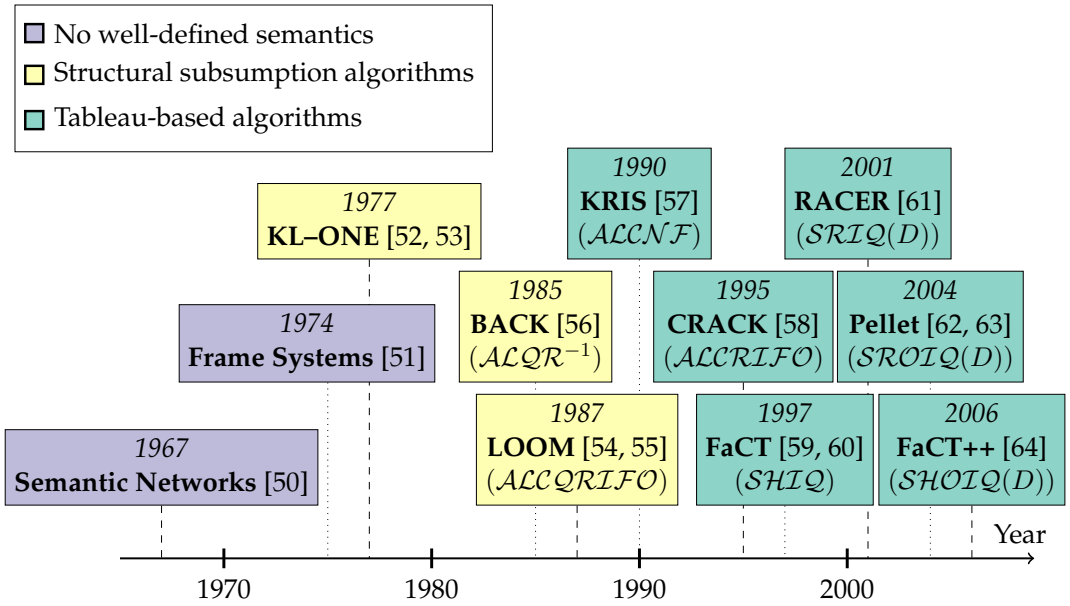


Figure 1: From semantic networks to modern Description Logic systems and reasoners.

Around 1977, the first Description Logic system KL-ONE [52, 53] appeared. It was based on the idea of structured inheritance networks by Brachman [67] and followed, like modern Description Logics, an open-world assumption. A new property of KL-ONE was that it, in contrast to its predecessors, could perform an automatic classification of new concepts by determining the closest concepts in terms of concept subsumption [68]. In Brachman and Levesque [69], the authors introduced a so-called *terminological component* and an *assertional component* to separate assertions about individuals from the concept definitions. This differentiation made its way into KRYPTON [70] and into later Description Logic systems. In the following time, the underlying semantics of these systems was explored which lead to the discovery of several unfortunate results concerning the employed structural subsumption algorithms [71, 72, 73]. These results included the finding that reasoning within expressive TBoxes is coNP-hard [73] and that the subsumption problem in KL-ONE and NIKL [74] is undecidable [75, 71]. A consequence of this was that the employed structural subsumption algorithms were sound but incomplete, i.e. they could not always detect all subsumption relationships. The following time was marked by the emerge of other systems like *LOOM* [54, 55] and *CLASSIC* [76] that tried to cope with these findings. The *CLASSIC* system for example was developed with a focus on a restriction of the expressive power in order to benefit reasoning [77].

In 1990, Schmidt-Schauß and Smolka [78] presented a tableau-based algorithm for the Description Logic $\mathcal{ALC}$. Following this work, a shift from structural subsumption algorithms to tableau-based algorithms occurred with the emerge of systems

like KRIS [57] and CRACK [58]. These were followed by optimized tableau-based systems like RACER [61] and FaCT [79, 59]. In addition, the link between Description Logics and other logics was explored. In 1991 it was shown by Schild [80] that the Description Logic $\mathcal{ALC}$ is closely related to the modal logic $K_{(m)}$ and in 1996 Borgida [81] showed that all Description Logics build from a set of common constructors correspond to a fragment of first-order predicate calculus. Results like this made it possible to transfer research results between the logics and led for instance to the extension of FaCT to satisfiability checking in modal logics [82]. In addition, other highly optimized systems like Racer [61] and Pellet [62, 63] emerged.

Since then, some work has been performed to enable Description Logics to deal with uncertainty [83, 84]. Examples for this include a combination of fuzzy logic and certain Description Logics such as $\mathcal{ALC}$ [85] and $\mathcal{SHOIN}(D)$ [86, 87]. The introduction of probabilistic extensions to Description Logics gave rise to logics like $\mathcal{ALCP}$ [88]. Moreover, other Description Logics were introduced particularly with a focus on the complexity of reasoning tasks. An example for this is the Description Logic family DL-Lite [89] where certain reasoning tasks, such as checking for concept and role subsumption relationships, can be performed in polynomial time.

A more in-depth summary of the history of Description Logic is provided by Baader et al. [90]. For current research on Description Logics the reader is referred to the International Workshop on Description Logics and its proceedings.[11]

### 3.2. Applications

Description Logic systems have been used in a variety of application areas such as information management [91], natural language processing [92], data mining [93], in the management of disasters [94] and in configuration systems used by Ford [95, 96] and AT&T [97, 98]. A large application area for Description Logics are ontology languages, especially OWL, where they provide the logical base: OWL 2 with direct model-theoretic semantics is closely related to the Description Logic $\mathcal{SROIQ}$ [99], OWL 2 EL corresponds to $\mathcal{EL}$++ [27, 28], OWL 2 QL to DL-Lite$_R$ [89] and OWL 2 RL to Description Logic Programs [100].[12] A lot of biomedical ontologies exist that employ OWL or correspond to certain Description Logics, such as the NCI Thesaurus [101] that contains vocabulary for cancer research and the aforementioned biomedical ontologies SNOMED CT and GO [12, 14]. The BioPortal [39, 40], which is a web portal provided by the National Center for Biomedical Ontology (NCBO), indexes close to 500 OWL ontologies at the time of writing. Moreover, a variety of other OWL ontologies can be found on the World Wide Web [102].

In the following section the Description Logic $\mathcal{EL}$ is introduced whose extension $\mathcal{EL}^{++}$ has found an application in the modeling of the biomedical domain and performs comparatively well in reasoning tasks.

[11] See http://dl.kr.org/workshops/ for an overview.
[12] See [26] for a more detailed overview of the OWL 2 profiles.

## 3.3. The Description Logic $\mathcal{EL}$

This section introduces the Description Logic $\mathcal{EL}$. After this, some additional definitions are given that include the TBox and the ABox of Description Logic systems. Following this, the *canonical extension* and *complete expansion* of a TBox are defined. This section then closes with an overview of common reasoning tasks in Description Logic systems and compares the complexity of reasoning in $\mathcal{EL}$ to other Description Logics.

### 3.3.1. Syntax and Semantics

In the following, the logic $\mathcal{EL}$ is defined based on Baader et al. [90]. The basic building blocks in $\mathcal{EL}$ are *atomic concepts* and *atomic roles*. The concepts in $\mathcal{EL}$ are build as follows: Let $A$ denote an atomic concept and let $R$ denote either an atomic role or the so-called universal role $u$. Let $C$ and $D$ denote concepts. Furthermore, let the set of the atomic concept names and the set of the atomic role names be disjoint. A concept $C'$ can then be built using the following syntax rule:

$$\begin{aligned} C' \leftarrow & \top \mid && \textit{universal concept} \\ & A \mid && \textit{atomic concept} \\ & C \sqcap D \mid && \textit{concept intersection} \\ & \exists R.C && \textit{existential quantification} \end{aligned}$$

The semantics of these concepts is then given by an *interpretation* $I$ that consists of a non-empty set $\Delta^I$, called the *domain*, and an interpretation function $\cdot^I$:

$$I = (\Delta^I, \cdot^I)$$

This interpretation function assigns every atomic concept $A$ a subset $A^I$ of the domain:

$$A^I \subseteq \Delta^I$$

The semantics of the top concept is given by:

$$\top^I = \Delta^I$$

The interpretation function assigns every role name $R$ a relation $R^I$ over the domain:

$$R^I \subseteq \Delta^I \times \Delta^I$$

Moreover, the semantics of the universal role is as follows:

$$u^I = \Delta^I \times \Delta^I$$

| Syntax | Semantics |
|---|---|
| $\top$ | $\Delta^I$ |
| $A$ | $A^I$ |
| $C \sqcap D$ | $C^I \cap D^I$ |
| $\exists R.C$ | $\{a \in \Delta^I \mid \exists b \in \Delta^I \textit{ with } (a,b) \in R^I \textit{ and } b \in C^I\}$ |

Table 1: The semantics of $\mathcal{EL}$.

In other words, the universal role connects all elements of the domain. In addition, the interpretation function maps every individual $a$ to an element $a^I$ of the domain:

$$a^I \in \Delta^I$$

Table 1 shows how the semantics is extended over complex concepts.
Consider now the following $\mathcal{EL}$ concept that could represent cats:

$$Animal \sqcap \exists PreysOn.Mouse$$

Concepts like this can be used to model a domain. For this, they are given a name[13] and aggregated within the so-called TBox. The TBox is a part of the knowledge base of Description Logic systems which is introduced in the following section. The next section further introduces other required definitions and explains the semantics of $\mathcal{EL}$ more in-detail.

### 3.3.2. Basic Definitions

In this section, some basic definitions are given that are mainly based on Baader and Nutt [103]. First, some definitions are stated that make it possible to express facts about concepts and axioms. Based on these definitions, the *TBox* and the *ABox* are defined which make up the knowledge base of a Description Logic system.

Let $C$ and $C'$ be concepts and let $A$, $B$ and $D$ be atomic concepts. Moreover, let $R$ and $S$ be roles. The definitions are then as follows:

**Definition 1** (Trivial conjunction)**.** *A conjunction of concepts is said to be trivial iff it consists of exactly one conjunct. It is said to be non-trivial iff it is not a trivial conjunction.*

**Definition 2** (Duplicate conjuncts)**.** *Let $C = C_1 \sqcap C_2 \sqcap \ldots \sqcap C_n$ be a concept such that each $C_i$ with $i \in \{1, 2, \ldots, n\}$ is a trivial conjunction. Then:*

$$C \textit{ contains duplicate conjuncts iff } \exists i, j \in \{1, 2, \ldots, n\} : i \neq j \textit{ and } C_i = C_j$$

*In other words, a concept is said to contain* duplicate conjuncts *iff it contains a conjunct that occurs more than once.*

[13] More formally, concept equivalence and concept subsumption are used to model relationships between concepts. They are introduced in the next section.

**Definition 3** (Symbols within a concept)**.** *The set $Atoms(C)$ is defined to contain all atomic concepts that occur within $C$, independent of the nesting depth of role restrictions these atomic concepts occur under. In addition, the set $Roles(C)$ is defined to contain all roles that occur within $C$, including the universal role, and independent of the number of role restrictions these roles occur under.*

An example for this definition is given in the following:

**Example 1** (Symbols within a concept)**.**

$$Atoms(A \sqcap \exists R.(B \sqcap D)) = \{A, B, D\}$$
$$Roles(A \sqcap \exists R.(B \sqcap \exists S.D)) = \{R, S\}$$

Some additional definitions are needed concerning the semantics of concepts and to express relationships between them. They are given in the following and are based on Baader et al. [104]:

**Definition 4** (Model of a concept)**.** *An interpretation $I$ is said to be a* model *of a concept $C$ iff $C^I \neq \emptyset$.*

**Definition 5** (Terminological axioms)**.** *Terminological axioms are used to express a relationship between concepts. These axioms consist of concept inclusion axioms, denoted by*

$$C \sqsubseteq C' \tag{1}$$

*and concept equivalence axioms, denoted by:*

$$C \equiv C' \tag{2}$$

*In these axioms, $C$ is said to occur* left-hand *whereas $C'$ is said to occur* right-hand*. In the case of a concept inclusion relationship the concept that occurs on the right-hand side of the axiom is said to* subsume *the concept on the left-hand side. In the example of* (1)*, $C'$ subsumes $C$. Concerning the semantics, an interpretation $I$ is a model of the concept inclusion axiom displayed in* (1) *iff:*

$$C^I \subseteq C'^I$$

*In addition, $I$ is a model of the concept equivalence axiom displayed in* (2) *iff:*

$$C^I = C'^I$$

*Based on the type of concept that occurs on the left-hand side of an axiom different types of concept relationship axioms are distinguished. These are as follows:*

***General concept inclusion (GCI):*** *Every concept inclusion axiom is a* general concept inclusion axiom*, i.e. a GCI axiom has the form:*

$$C \sqsubseteq C'$$

***Concept specialization:*** *A* concept specialization*, sometimes also called* primitive concept definition *[104], is a concept inclusion axiom whose left-hand side is an atomic concept:*

$$A \sqsubseteq C$$

***Concept definition:*** *A* concept definition *is a concept equivalence axiom whose left-hand side is an atomic concept:*

$$A \equiv C$$

**Example 2** (Terminological axioms)**.** *Consider the following two terminological axioms:*

$$Mouse \sqsubseteq Mammal \tag{3}$$

$$Cat \equiv Animal \sqcap \exists PreysOn.Mouse \tag{4}$$

*The axiom displayed in* (3) *states that every mouse is a mammal and the axiom shown in* (4) *states that every cat is an animal that preys on mice.*

Axioms like these can be used to conceptualize a domain. For this, they are aggregated within the so-called TBox which is a part of the knowledge base of Description Logic systems. The TBox is defined in the following:

**Definition 6** (Syntax and semantics of the TBox)**.** *The* TBox *is the part of a knowledge base that contains terminological knowledge about a domain. Different types of TBoxes are distinguished depending on their content:*

***Generalized TBox:*** *A* generalized TBox *consists of a finite set of concept definitions and concept specializations. Only a single axiom per definition is allowed, i.e. for every atomic concept $A$ in a TBox $\mathcal{T}$ there exists at most one axiom in $\mathcal{T}$ where $A$ occurs left-hand.*

***Normalized TBox:*** *A* normalized TBox *contains a finite set of concept definitions. For every atomic concept $A$ there is at most one concept definition where $A$ occurs left-hand. In addition, a normalized TBox is called* acyclic *iff no definition contains any direct or indirect cycles.*[14]

*Concerning the semantics of a TBox, an interpretation $I$ is a model of a TBox iff it is a model for every axiom in that TBox.*

**Example 3** (TBox)**.** *Consider again the axioms* (3) *and* (4) *from the previous example. A TBox that contains both axioms is a generalized TBox whereas a TBox that only contains the axiom* (4) *is a normalized TBox. Moreover, the latter TBox is also acyclic.*

A more extensive example for a TBox is given at the end of this section. Before this, some additional definitions concerning the concepts within a TBox are made.

[14] An example for a terminological axiom that contains a direct cycle is $A \equiv \exists R.A$.

**Definition 7** (Unsatisfiable concept w.r.t. a TBox)**.** *A concept $C$ is said to be* unsatisfiable *w.r.t. a TBox $\mathcal{T}$ iff for all models $I$ of $\mathcal{T}$ it is implied that $C^I = \emptyset$.*

**Definition 8** (Incoherent TBox)**.** *A TBox $\mathcal{T}$ is called* incoherent *iff $\mathcal{T}$ contains a concept that is unsatisfiable w.r.t. $\mathcal{T}$.*[15]

**Definition 9** (Consequence of a TBox)**.** *Let $\mathcal{T}$ be a TBox and let $\phi$ denote a concept relationship between two concepts. $\phi$ is said to be a* consequence of *$\mathcal{T}$ iff all models of $\mathcal{T}$ are also models of $\phi$. To denote such a consequence one writes:*

$$\mathcal{T} \models \phi$$

**Example 4** (Consequence of a TBox)**.** *To express that a given TBox $\mathcal{T}$ entails the consequence that $Animal$ subsumes $Cat$, one can write:*

$$\mathcal{T} \models Cat \sqsubseteq Animal$$

**Definition 10.** *Let $\mathcal{T}$ be a TBox. Then the following definitions are made:*

***Set $\mathcal{N}_\mathcal{T}$ of name symbols:*** *The* set $\mathcal{N}_\mathcal{T}$ of name symbols *of $\mathcal{T}$ is defined to consist of all atomic concepts that occur left-hand of a terminological axiom in $\mathcal{T}$.*

***Named concept:*** *The term* named concept *is used in the following to express that a concept is a member of the set $\mathcal{N}_\mathcal{T}$, i.e.:*

$$C \text{ is a named concept in } \mathcal{T} \text{ iff } C \in \mathcal{N}_\mathcal{T}$$

***Set $\mathcal{B}_\mathcal{T}$ of base symbols:*** *The* set $\mathcal{B}_\mathcal{T}$ of base symbols *is defined to contain all atomic concepts that occur solely on the right-hand side of all terminological axioms in $\mathcal{T}$.*

***Set $\mathcal{R}_\mathcal{T}$ of role names:*** *The* set $\mathcal{R}_\mathcal{T}$ of role names *is defined to contain all role names that occur within concepts in $\mathcal{T}$, including the universal role $u$.*

**Definition 11** (Syntax and semantics of an ABox)**.** *The* ABox *of a Description Logic knowledge base contains assertional knowledge about the domain. It comprises of a finite set of* concept assertions *and* role assertions*:*

***Concept assertion:*** *A concept assertion assigns an individual $a$ to a concept $C$, written as $a\colon C$. An interpretation $I$ is a model of the concept assertion $a\colon C$ iff $a^I \in C^I$.*

***Role assertion:*** *A role assertion expresses the fact that an individual $b$ is in the filler of a role $R$ w.r.t. another individual $a$, written as $(a, b)\colon R$. An interpretation $I$ is a model of the role assertion $(a, b)\colon R$ iff $(a^I, b^I) \in R^I$.*

*Concerning the semantics, $I$ is a model of an ABox $\mathcal{A}$ iff $I$ is a model for all axioms in $\mathcal{A}$.*

[15] See also Schlobach et al. [105].

**Example 5** (Assertions). *Consider again the concepts from Example 2:*

$$\begin{aligned} Mouse &\sqsubseteq Mammal \\ Cat &\equiv Animal \sqcap \exists PreysOn.Mouse \end{aligned}$$

*Examples for related concept assertions are:*

$$\begin{aligned} jerry &: Mouse \\ tom &: Cat \end{aligned}$$

*An example for a role assertion is given by:*

$$(tom, jerry) : PreysOn$$

Together, the ABox and the TBox make up the knowledge base of Description Logic systems.

**Definition 12** (Knowledge base of a Description Logic system). *The* knowledge base *of a Description Logic system consists of a TBox and an ABox. In other words, let $\mathcal{T}$ denote a TBox and let $\mathcal{A}$ denote an ABox. Then a knowledge base $\mathcal{K}$ that consists of this TBox and ABox is given by the pair:*

$$\mathcal{K} = (\mathcal{T}, \mathcal{A})$$

*An interpretation $I$ is said to be a model of $\mathcal{K}$ iff it is a model for both $\mathcal{T}$ and $\mathcal{A}$.*

The following example aims at clarifying the previous definitions.

**Example 6** (Definitions). *Consider the following set of axioms:*

$$Cat \equiv Animal \sqcap \exists PreysOn.Mouse \tag{5}$$

$$Mouse \equiv Animal \sqcap \exists hasPreference.Cheese \tag{6}$$

$$Cheese \equiv Milk \sqcap \exists hasTreatment.Coagulation \tag{7}$$

*For reasons of simplification, it is assumed that cheese can be modeled as a special form of milk that has undergone a treatment that lead to its coagulation.*[16] *The atomic concepts that occur in these axioms are given by the following set:*

$$\{Cat, Animal, Mouse, Milk, Cheese, Coagulation\}$$

*In addition, all concepts that occur on the left-hand side are atomic concepts and only occur once. The axioms* (5)–(7) *therefore make up a TBox $\mathcal{T}$. $\mathcal{T}$ is a normalized TBox, because it contains no concept specialization. Moreover, $\mathcal{T}$ is acyclic, because no concept definition*

[16] In reality, the process is more complex and involves the separation of the solid part of the coagulated milk from the whey.

*contains a direct or indirect cycle. The concepts $Cat$, $Mouse$ and $Cheese$ are named concepts, because they occur on the left-hand side of a concept definition. Since there are no other named concepts, they make up the set $\mathcal{N}_\mathcal{T}$:*

$$\mathcal{N}_\mathcal{T} = \{Cat, Mouse, Cheese\}$$

*The set of base symbols on the other hand consists of all symbols that occur only on the right-hand side of all concept definitions:*

$$\mathcal{B}_\mathcal{T} = \{Animal, Coagulation, Milk\}$$

*The set of role names contains all role names that appear within the TBox:*

$$\mathcal{R}_\mathcal{T} = \{PreysOn, hasPreference, hasTreatment\}$$

*Now consider the following assertional axioms:*

$$tom : Animal \tag{8}$$
$$jerry : Animal \tag{9}$$
$$tom : Cat \tag{10}$$
$$jerry : Mouse \tag{11}$$
$$berkswell : Cheese \tag{12}$$
$$gouda : Cheese \tag{13}$$
$$sheepmilk : Milk \tag{14}$$
$$cowmilk : Milk \tag{15}$$
$$gouda : Milk \tag{16}$$
$$berkswell : Milk \tag{17}$$
$$rennetbasedcoagulation : Coagulation \tag{18}$$
$$(tom, jerry) : PreysOn \tag{19}$$
$$(jerry, gouda) : hasPreference \tag{20}$$
$$(jerry, berkswell) : hasPreference \tag{21}$$
$$(gouda, rennetbasedcoagulation) : hasTreatment \tag{22}$$
$$(berkswell, rennetbasedcoagulation) : hasTreatment \tag{23}$$

*These axioms make up an ABox, which is denoted by $\mathcal{A}$ in the following. Together, the TBox and the ABox form a knowledge base $\mathcal{K}$:*

$$\mathcal{K} = (\mathcal{T}, \mathcal{A})$$

*Now consider the following interpretation $I$:*

$$\begin{aligned}
\Delta^I &= \{tom, jerry, gouda, berkswell, \\
&\quad sheepmilk, cowmilk, rennetbasedcoagulation\} \\
Animal^I &= \{tom, jerry\} \\
Cat^I &= \{tom\} \\
Mouse^I &= \{jerry\} \\
Cheese^I &= \{gouda, berkswell\} \\
Milk^I &= \{sheepmilk, cowmilk, gouda, berkswell\} \\
Coagulation^I &= \{rennetbasedcoagulation\} \\
PreysOn^I &= \{(tom, jerry)\} \\
hasPreference^I &= \{(jerry, gouda), (jerry, berkswell)\} \\
hasTreatment^I &= \{(gouda, rennetbasedcoagulation), \\
&\quad (berkswell, rennetbasedcoagulation)\} \\
a^I &= \ a \ for\ all\ individuals\ a\ in\ \mathcal{A}
\end{aligned}$$

*This interpretation satisfies all the ABox axioms* (8)–(23) *and is therefore a model of the ABox. Concerning the TBox, consider again the definition of $Cheese$ in axiom* (7)*:*

$$Cheese \equiv Milk \sqcap \exists hasTreatment.Coagulation$$

*$I$ is a model of this axiom if and only if:*

$$Cheese^I = (Milk \sqcap \exists hasTreatment.Coagulation)^I \tag{24}$$

*From the definition of $I$ it follows that:*

$$Cheese^I = \{gouda, berkswell\} \tag{25}$$

*It is now shown that this is equivalent to the right-hand side of* (24)*:*

$$\begin{aligned}
&(Milk \sqcap \exists hasTreatment.Coagulation)^I \\
\equiv\ &Milk^I \cap (\exists hasTreatment.Coagulation)^I \\
\equiv\ &\{cowmilk, sheepmilk, gouda, berkswell\} \\
&\cap \{a \in \Delta^I \mid \exists b \in \Delta^I \text{ with } (a, b) \in hasTreatment^I \text{ and } b \in Coagulation^I\} \\
\equiv\ &\{cowmilk, sheepmilk, gouda, berkswell\} \cap \{gouda, berkswell\} \\
\equiv\ &\{gouda, berkswell\}
\end{aligned}$$

*This equals* (25)*. It therefore follows that $I$ is a model of axiom* (7)*. In the same way, $I$ is a model of the concept definitions of $Cat$ and $Mouse$, too. But this means that $I$ satisfies all the axioms in $\mathcal{T}$. $I$ is therefore a model of $\mathcal{T}$ and thus also for the knowledge base $\mathcal{K}$.*

*Moreover, from set theory it follows that for every interpretation $J$ that is a model of axiom* (7)*, $Cheese^J$ only contains elements which are also elements of:*

$$(\exists hasTreatment.Coagulation)^J$$

*In other words, for all interpretations $J$ which are models of axiom* (7) *it is implied that:*

$$Cheese^J \subseteq (\exists hasTreatment.Coagulation)^J$$

*Since every model of $\mathcal{T}$ must also be a model of axiom* (7)*, it can be concluded that the equation above is true for all models of $\mathcal{T}$. It is therefore a consequence of the TBox:*

$$\mathcal{T} \models Cheese \sqsubseteq \exists hasTreatment.Coagulation$$

In the following section the canonical extension and complete expansion of a TBox are introduced.

### 3.3.3. Canonical Extension and Complete Expansion

In this section, the canonical extension and the complete expansion of a normalized TBox are defined. The definitions are based on Nebel [73].

A normalized TBox $\mathcal{T}$ can be seen as a function that assigns every name symbol the right-hand side of its definition. More formally, let $\mathcal{L}$ denote the set of all $\mathcal{EL}$ concepts. Then $\mathcal{T} : \mathcal{L} \to \mathcal{L}$ is defined as:

$$\mathcal{T}(C) = \begin{cases} D & \text{if } C \in \mathcal{N}_\mathcal{T} \text{ and } D \text{ is the right-hand side of the definition of } C \text{ in } \mathcal{T} \\ C & otherwise \end{cases}$$

For convenience reasons, $\mathcal{T}$ is defined to return the unchanged argument if it is not an element of $\mathcal{N}_\mathcal{T}$. The function $\mathcal{T}$ is then used in the following to define the *canonical extension* and the *complete expansion* of a concept w.r.t. $\mathcal{T}$:

**Definition 13** (Canonical extension $\hat{\mathcal{T}}(C)$)**.** *Let $\mathcal{T}$ be a normalized TBox. Then the* canonical extension *$C'$ of a concept $C$ w.r.t. $\mathcal{T}$ is obtained from $\mathcal{T}$ by replacing all name symbols $A \in \mathcal{N}_\mathcal{T}$ that occur in $C$ with $\mathcal{T}(A)$. More formally, for all concepts $C$*

$$C' = \hat{\mathcal{T}}(C)$$

*with $\hat{\mathcal{T}}$ defined as follows:*

$$\begin{aligned} \hat{\mathcal{T}}(B) &= B && \text{for } B \in \mathcal{B}_\mathcal{T} \\ \hat{\mathcal{T}}(A) &= \mathcal{T}(A) && \text{for } A \in \mathcal{N}_\mathcal{T} \\ \hat{\mathcal{T}}(\exists R.C_1) &= \exists R.\hat{\mathcal{T}}(C_1) && \text{for a role } R \text{ and a concept } C_1 \\ \hat{\mathcal{T}}(C_1 \sqcap C_2 \sqcap \ldots \sqcap C_n) &= \prod_{i=1}^{n} \hat{\mathcal{T}}(C_i) && \text{for } n > 1 \text{ and concepts } C_1, C_2, \ldots, C_n \end{aligned}$$

*In addition, let the $i$th canonical extension $\hat{\mathcal{T}}_i$ of a concept $C$ w.r.t. $\mathcal{T}$ be defined as:*

$$\hat{\mathcal{T}}_i(C) = \underbrace{\hat{\mathcal{T}} \circ \hat{\mathcal{T}} \circ \ldots \circ \hat{\mathcal{T}}}_{i \text{ times}}(C)$$

The complete expansion of a concept $C$ w.r.t. an acyclic TBox is then defined as follows:

**Definition 14** (Complete expansion $\mathcal{T}^*(C)$)**.** *Let $\mathcal{T}$ be a normalized TBox that is acyclic. Let the* depth *of $C$ w.r.t. $\mathcal{T}$ be defined as follows:*

$$depth(C) = \min(\{i \mid i \in \mathbb{N}\backslash\{0\} \text{ and } \hat{\mathcal{T}}_i(C) = \hat{\mathcal{T}}_{i+1}(C)\})$$

*Then the* complete expansion $\mathcal{T}^*(C)$ *of a concept $C$ w.r.t. $\mathcal{T}$ is defined as follows:*

$$\mathcal{T}^*(C) = \hat{\mathcal{T}}_{depth(C)}(C)$$

*In addition, if the depth of every concept in a TBox equals one, then this TBox is said to be completely expanded.*

In other words, the complete expansion $\mathcal{T}^*$ of a concept $C$ w.r.t. a TBox $\mathcal{T}$ can be computed by exhaustively computing the canonical extension of $C$ until its definition contains only concepts that are built from base symbols. If one replaces the definitions of all concepts of a TBox with their complete expansion, the resulting TBox $\mathcal{T}'$ is called completely expanded. Following Baader and Nutt [103], $\mathcal{T}$ and $\mathcal{T}'$ are equivalent and share the same name and base symbols. In the following an example for these definitions is given:

**Example 7** (Canonical extension and complete expansion)**.** *Consider again the TBox shown in Example 6:*

$$\begin{aligned} Cat &\equiv Animal \sqcap \exists PreysOn.Mouse \\ Mouse &\equiv Animal \sqcap \exists hasPreference.Cheese \\ Cheese &\equiv Milk \sqcap \exists hasTreatment.Coagulation \end{aligned}$$

*First note that because $Cat$, $Mouse$ and $Cheese$ are members of $\mathcal{N}_\mathcal{T}$, one can obtain the right-hand side of their definition using the function $\mathcal{T}$. This gives the same result as the first canonical extension:*

$$\begin{aligned} \hat{\mathcal{T}}(Cat) = \mathcal{T}(Cat) &= Animal \sqcap \exists PreysOn.Mouse \\ \hat{\mathcal{T}}(Mouse) = \mathcal{T}(Mouse) &= Animal \sqcap \exists PreysOn.Cheese \\ \hat{\mathcal{T}}(Cheese) = \mathcal{T}(Cheese) &= Milk \sqcap \exists hasTreatment.Coagulation \end{aligned}$$

*Note that the concept Cat contains the two concepts Mouse and $Animal$ within its definition. Since Mouse is a named symbol, the second canonical extension of $Cat$ differs from*

*the first one in that Mouse is replaced by $\mathcal{T}(Mouse)$. $Animal$ on the other hand is a base symbol and is therefore not replaced:*

$$\begin{aligned}\hat{\mathcal{T}}_2(Cat) &= \hat{\mathcal{T}}(\hat{\mathcal{T}}(Cat)) \\ &= \hat{\mathcal{T}}(Animal \sqcap \exists PreysOn.Mouse) \\ &= \hat{\mathcal{T}}(Animal) \sqcap \hat{\mathcal{T}}(\exists PreysOn.Mouse) \\ &= Animal \sqcap \exists PreysOn.\hat{\mathcal{T}}(Mouse) \\ &= Animal \sqcap \exists PreysOn.\mathcal{T}(Mouse) \\ &= Animal \sqcap \exists PreysOn.(Animal \sqcap \exists hasPreference.Cheese)\end{aligned}$$

*Since $\hat{\mathcal{T}}_2(Cat)$ still contains a named symbol, namely $Cheese$, the third canonical extension of $Cat$ returns a different result in which the concept $Cheese$ is replaced by the right-hand side of its definition:*

$$\begin{aligned}\hat{\mathcal{T}}_3(Cat) =& \hat{\mathcal{T}}(\hat{\mathcal{T}}(\hat{\mathcal{T}}(Cat))) \\ =& \hat{\mathcal{T}}(Animal \sqcap \exists PreysOn.(Animal \sqcap \exists hasPreference.Cheese)) \\ =& \hat{\mathcal{T}}(Animal) \sqcap \hat{\mathcal{T}}(\exists PreysOn.(Animal \sqcap \exists hasPreference.Cheese)) \\ =& Animal \sqcap \exists PreysOn.\hat{\mathcal{T}}(Animal \sqcap \exists hasPreference.Cheese) \\ =& Animal \sqcap \exists PreysOn.(\hat{\mathcal{T}}(Animal) \sqcap \hat{\mathcal{T}}(\exists hasPreference.Cheese)) \\ =& Animal \sqcap \exists PreysOn.(Animal \sqcap \exists hasPreference.\hat{\mathcal{T}}(Cheese)) \\ =& Animal \sqcap \exists PreysOn.(Animal \sqcap \exists hasPreference.\mathcal{T}(Cheese)) \\ =& Animal \sqcap \exists PreysOn.(Animal \sqcap \exists hasPreference.(Milk \sqcap \\ & \exists hasTreatment.Coagulation))\end{aligned}$$

*Since no more named symbols are present, the fourth canonical extension equals the third one. The depth of the concept $Cat$ is therefore three:*

$$depth(Cat) = 3$$

*The complete expansion of $Cat$ is subsequently given by:*

$$\begin{aligned}\mathcal{T}^*(Cat) &= \hat{\mathcal{T}}_{depth(Cat)}(Cat) \\ &= \hat{\mathcal{T}}_3(Cat) \\ &= Animal \sqcap \exists PreysOn.(Animal \sqcap \exists hasPreference.(Milk \sqcap \\ &\quad \exists hasTreatment.Coagulation))\end{aligned}$$

The next section gives an overview of common reasoning tasks in Description Logics.

### 3.3.4. Reasoning

The complexity of reasoning is an important aspect of Description Logics and ontology languages. The OWL profiles have been intentionally designed in such a way that the basic reasoning tasks are decidable [26]. Speaking of Description logics, the main reasoning tasks w.r.t. the TBox are concept *classification* and determining *logical implications* [106]. Classification requires the computation of the hierarchy of concept subsumption relationships and determines the closest super and sub concepts of a given concept. Logical implication tests are used to determine if a given relationship between concepts is a consequence of the TBox. The main reasoning task w.r.t. the ABox on the other hand is *instance checking* [106]. Instance checking is used to determine if an individual is an instance of a concept. Other important reasoning tasks can be reduced to it, such as checking the *consistency* of a knowledge base by testing if every concept has a model.

The complexity of these reasoning tasks depends on the Description Logic. The logic $\mathcal{EL}$ and some of its extension behave well in these tasks: The classification problem in cyclic $\mathcal{EL}$ terminologies for example can be solved in polynomial time [107, 108]. This is still true if general concept inclusions are allowed [29]. Brandt [29] also showed that this complexity does not change even if $\mathcal{EL}$ is extended by simple role inclusion of the form

$$R \sqsubseteq S$$

where $R$ and $S$ are roles. The resulting language is called $\mathcal{ELH}$. Furthermore, Baader et al. [27, 28] added the bottom concept $\bot$, nominals and role inclusions of the form

$$R_1 \circ R_2 \circ \ldots \circ R_n \sqsubseteq S$$

to $\mathcal{EL}$, where $R_1$, $R_2$, ..., $R_n$ are roles. These role inclusions are a generalization of simple role inclusions. By adding these constructors to $\mathcal{EL}$ one obtains the language $\mathcal{EL}^{++}$. Baader et al. [27, 28] then showed that despite these new constructors the subsumption problem in $\mathcal{EL}^{++}$ remains tractable.

These complexity results stand in sharp contrast to the reasoning complexity of many other Description Logics. Adding universal quantification to $\mathcal{EL}$ results in the logic $\mathcal{ALE}$, where the concept subsumption problem is NP-complete. In the logic $\mathcal{ALC}$, subsumption is PSPACE-complete [109]. Reasoning in the very expressive logic $\mathcal{SROIQ}$, a logic that is closely related to OWL 2 with direct model-theoretic semantics, turned out to be N2ExpTime-complete [110].

Today a variety of highly optimized Description Logic reasoners exists [63, 111, 112, 113] that have to compete with each other in these reasoning tasks.[17] Some of them provide only support for certain OWL profiles, such as *ELK* [113], which is a reasoner that only supports the profile OWL 2 EL.

[17] See Parsia et al. [114] for a benchmark of reasoners that used OWL 2 EL and OWL 2 DL ontologies.

### 3.4. Conclusion

In this section, the history of Description Logic was summarized and application areas for this logic were shown. The Description Logic $\mathcal{EL}$ was introduced and some basic definitions were given. Following this, the canonical extension and the complete expansion of a TBox were defined. Common reasoning tasks for Description Logics were listed and it was noted that the subsumption problem in $\mathcal{EL}$ and some of its extensions is tractable. Based on this and the fact that $\mathcal{EL}$ and its extension have a practical relevance, this logic is chosen as the base for the development of the knowledge retraction operator. The next section gives an overview of related work.

## 4. Related Work

This section examines related work. First, the AGM postulates for belief set contraction and the postulates for belief base contraction are introduced. After this, methods are analyzed that are concerned with the debugging of ontologies and Description Logic knowledge bases. In a next step, work that deals with determining the difference between concepts is examined and its applicability to the problem of a fine-grained removal of knowledge from concepts is judged.

### 4.1. AGM Belief Change

The AGM postulates [5], named after their inventors Alchourrón, Gärdenfors and Makinson, state general properties that an operator that changes the believes of a system should follow. Their work includes postulates for the removal of knowledge (*belief contraction*), the addition of new knowledge (*belief expansion*) and the revision of existing knowledge (*belief revision*). The latter also deals with removing inconsistencies that can occur after the incorporation of new knowledge into the system. The AGM postulates are formulated independent of the actual language that is used to represent the knowledge. They are based on a set of propositions that expresses one's believes. More formally, a *belief set* is a set $K$ of propositions that is closed under the consequence relation $Cn$, i.e.:

$$K \textit{ is a belief set iff } K = Cn(K)$$

In the following, only the postulates for belief contraction are of interest. Let $K$ denote a belief set, $Cn(K)$ the deductive closure of $K$ and $\phi$ a logical sentence. The six basic AGM postulates for belief set contraction are then shown in Table 2. Each postulate covers a different aspects of the contraction operation. The *closure* postulate ensures that the result of the operation is again a belief set and the *inclusion* postulate ensures that no new knowledge is gained when knowledge is removed. The *vacuity* postulate is arranged in such a way that knowledge is only removed from a belief set if it is currently present. If this is not the case, then nothing is changed. *Success* ensures that the operation can be trusted in that the knowledge which is sought to

| Postulate | Definition |
|---|---|
| Closure | $K - \phi$ is a belief set |
| Inclusion | $K - \phi \subseteq K$ |
| Vacuity | If $\phi \notin Cn(K)$, then $K - \phi = K$ |
| Success | If $\phi \notin Cn(\emptyset)$, then $\phi \notin Cn(K - \phi)$ |
| Preservation | If $Cn(\phi) = Cn(\phi')$, then $K - \phi = K - \phi'$ |
| Recovery | $K \subseteq Cn((K - \phi) \cup \phi)$ |

Table 2: The six basic AGM postulates for belief set contraction.

be removed is not present in the result anymore. The *preservation* postulate on the other hand ensures that the contraction of semantically equivalent sentences gives the same result. Last, the *recovery* postulate enforces some minimality of the contraction: adding back the sentence that was contracted must be sufficient to restore all the previously present knowledge. Note that this postulate is the only one that limits the removal of knowledge.[18] For a more in-depth explanation of these postulates the reader is referred to Alchourrón et al. [5].

Besides performing a contraction on belief sets, one can also perform the contraction on *belief bases* [4, 115]. These are sets of logical sentences which are not necessarily closed under logical consequence. The name stems from the fact that they provide an (usually finite) base for a belief set. They are defined as follows:

$$B \textit{ is a belief base for a belief set } K \text{ iff } K = Cn(B)$$

Hansson [4, 115] lists some basic postulates that an operator for belief base contraction should fulfill. These postulates include, amongst others, the ones displayed in Table 3. Note that inclusion, vacuity and success are quite similar to the corresponding postulates for belief set contraction modulo their definition on belief bases. Note further that the relevance postulate is related to the recovery postulate. From the postulates shown in Table 3 only inclusion, relevance, success and uniformity are needed as postulates. Vacuity is usually omitted, because it can be inferred from these postulates [115].

The application of the recovery and relevance postulates to the logic $\mathcal{EL}$ are examined in the following. Since the aim of this thesis is the development of an operator for retracting knowledge from concepts, the postulates will be analyzed for their applicability at concept level. Consider both postulates again:

**Recovery:** $K \subseteq Cn((K - \phi) \cup \phi)$

**Relevance:** If $\beta \in B$ and $\beta \notin B - \phi$, then there exists a set $B'$ such that $B - \phi \subseteq B' \subseteq B$ and $\phi \notin Cn(B')$ but $\phi \in Cn(B' \cup \{\beta\})$

[18] See Section 6.2.2 for more details on this.

| Postulate | Definition |
|---|---|
| Inclusion | $B - \phi \subseteq B$ |
| Vacuity | If $\phi \notin Cn(K)$, then $K - \phi = K$ |
| Success | If $\phi \notin Cn(\emptyset)$, then $\phi \notin Cn(B - \phi)$ |
| Uniformity | If for all $B' \subseteq B$: $\phi \cap Cn(B') = \emptyset$ iff $\phi' \cap Cn(B') = \emptyset$ then: $B - \phi = B - \phi'$ |
| Relevance | If $\beta \in B$ and $\beta \notin B - \phi$, then there exists a set $B'$ such that $B - \phi \subseteq B' \subseteq B$ and $\phi \notin Cn(B')$ but $\phi \in Cn(B' \cup \{\beta\})$ |

Table 3: Some of the postulates for belief base contraction.

Recovery states that if a belief was removed from a belief set and is added back, the resulting belief set must again contain all the previous beliefs. Relevance states that if a belief $\beta$ was present in a belief base and has been removed when some belief $\phi$ was retracted, then it must have played some role in the entailment of $\phi$. Consequently, there must exist a belief base between $B$ and $B - \phi$ such that adding back $\beta$ is sufficient to restore $\phi$. Note that both of these postulates need an operation to restore a lost entailment. This poses a problem in the application of these postulates to $\mathcal{EL}$ concepts. In the case of the recovery postulate, it is easy to see that naïvely adding back the lost belief is not possible. To give an example, assume there exists an operator $\ominus$ that can be used to retract a consequence from an $\mathcal{EL}$ concept. Then consider the following example result of a retraction operation:

$$\exists R.(A \sqcap B) \ominus \exists R.A = \exists R.B$$

One could try to add back $\exists R.A$ to the result $\exists R.B$ using concept intersection. However, the result would differ from the minuend:

$$\exists R.B \sqcap \exists R.A \not\sqsubseteq \exists R.(A \sqcap B)$$

It is therefore not possible to naïvely employ the AGM recovery postulate at concept level. Since the relevance postulate also needs an operation to restore a lost belief, the same problem arises here.

In this section, the postulates for belief base contraction and belief set contraction were introduced. The application of the recovery and relevance postulates at concept level was examined and it was found that these postulates cannot be applied naïvely. The next section analyzes approaches that are concerned with the debugging of ontologies and the determination of the difference between Description Logic concepts.

## 4.2. Debugging Ontologies and Description Logic Knowledge Bases

Debugging and repairing inconsistent ontologies is an active area of research [116, 117, 118, 119, 120, 121]. It led to the development of tools such as Pellet [63] and the NeOn[19] plugin RADON [122] that can assist human ontology editors in explaining inferences and computing repairs. Debugging an ontology includes to find the cause of an unwanted entailment. An inconsistency in an ontology for example can be caused by an incoherent TBox $\mathcal{T}$. This means that the ontology contains an unsatisfiable concept. This incoherency in the TBox can be understood as the entailment $\mathcal{T} \models \bot$. Removing this entailment therefore also resolves the incoherency. In this section, work that deals with finding the reasons for inconsistencies and resolving them is examined in order to judge its applicability to the task of retracting knowledge from concepts. The first part introduces (laconic) justifications. These can be used to identify axioms and axiom parts that are responsible for the entailment of a consequence. The second part then examines research that deals with the resolution of inconsistencies at a fine-grained level and shows why these approaches are not immediately applicable to $\mathcal{EL}$.

### 4.2.1. Justifications

This section introduces *justifications* and *laconic justifications*. Assume in the following that one wants to retract an unwanted consequence from an ontology. In this case it could happen that there exists more than one axiom or more than one combination of axioms which cause the entailment of this consequence. Consider for example the following axioms:

$$A \sqsubseteq B \sqcap D \tag{26}$$
$$A \sqsubseteq C \sqcap E \tag{27}$$
$$C \sqsubseteq B \tag{28}$$

If these axioms were part of the TBox $\mathcal{T}$ of an ontology, one could infer that:

$$\mathcal{T} \models A \sqsubseteq B \tag{29}$$

If asked why this consequence is entailed, one could rightfully refer to either axiom (26) or the combination of the axioms (27) and (28). This issue is captured by the concept of *justifications* [123, 124]. Informally, every minimal set of axioms that entails a given consequence is called a justification w.r.t. this consequence. After computing all justifications for a consequence, for example by using a technique called *axiom pinpointing* [120], they can be used to retract this consequence from the ontology. This can be done by simply removing at least one axiom per justification from the ontology. Using the previous example, the two justifications for the consequence

[19] See http://www.neon-project.org.

displayed in (29) are:

$$J_1 = \{A \sqsubseteq B \sqcap D\}$$
$$J_2 = \{A \sqsubseteq C \sqcap E, C \sqsubseteq B\}$$

On the other hand, the set

$$\{A \sqsubseteq B \sqcap D, A \sqsubseteq C \sqcap E, C \sqsubseteq B\}$$

is not a justification, because it is not minimal.

Not every part of an axiom that belongs to a justification for a consequence is necessarily responsible for the entailment of this consequence. Consider again axiom (27). Only the first conjunct of (27) is responsible for its presence in the justification $J_2$. If this axiom was to be removed, one would also remove the fact

$$A \sqsubseteq E$$

from the ontology. To cope with this, Horridge et al. [7] defined so-called *laconic justifications*. Informally, a laconic justification for a consequence is a justification which is stripped of all parts that are not relevant for the entailment of this consequence. More detailed, to decide if a set $J$ of axioms is a laconic justification for a consequence, a satisfiability-preserving structural transformation based on Plaisted and Greenbaum [125] is applied. This transformation splits each axiom into a set of flattened axioms. If these axioms contain no parts that are unnecessary in the entailment of the consequence or can be weakened, then $J$ is a laconic justification.[20] To give an example, consider again the axioms (26) – (28). A laconic justification for the consequence displayed in (29) w.r.t. these axioms is:

$$\{A \sqsubseteq C, C \sqsubseteq B\}$$

It stems from the axioms (27) and (28).[21]

Since laconic justifications provide the means to identify the problematic parts of an axiom, they can prove useful in the process of a fine-grained removal of consequences from concepts. The next section examines other work that deals with resolving inconsistencies at a fine-grained level.

#### 4.2.2. Resolving Inconsistencies at a Fine-grained Level

This section lists some research that deals with resolving inconsistencies at a fine-grained level. Schlobach and Cornet [120] introduced the method of *concept pinpointing* in the logic $\mathcal{ALC}$. Given an incoherent TBox $\mathcal{T}$, they use this method to

[20] In theory, the possible candidates for laconic justifications are obtained from the deductive closure of the ontology. In practice, a filter is applied before the candidates are extracted.

[21] Note that both elements of this justification are present in the deductive closure of an ontology that consists of the TBox axioms $\{A \sqsubseteq B \sqcap D, A \sqsubseteq C \sqcap E, C \sqsubseteq B\}$.

identify the parts of axioms that cause this incoherency. For this, they derive a set of new TBoxes from $\mathcal{T}$ by changing the axioms in $\mathcal{T}$ while preserving the incoherency. To achieve this, their algorithm replaces the conjuncts in the axioms with syntactically related but more general conjuncts in terms of concept subsumption. They call this process the *weakening* of conjuncts and argue that axiom parts that can only be weakened to a certain degree are a possible cause of the incoherency.

Lam et al. [126] propose a method that is related to Schlobach and Cornet [120] and builds upon the tableau algorithm by Meyer et al. [127]. This method also identifies the parts of the axioms that are responsible for an inconsistency. For these parts they compute possible repairs which are split in so-called *helpful* and *harmful* changes. This splitting is based on the impact that applying these repairs would have on the entailment of other, unrelated consequences. They developed a plugin for the ontology editor Protégé that assists the users in computing these helpful changes.

Besides the work listed here, other work exists that deals with debugging inconsistent ontologies [121]. In general, all these approaches could be extended to the task of removing a consequence $P$ from a concept $C$. This could be done by introducing an incoherency into the ontology. Given that

$$C \sqsubseteq P$$

one could temporarily add the axiom:

$$C \sqsubseteq \neg P \tag{30}$$

This could then be followed by the computation of changes that can be applied to the ontology to recover from this incoherency. These changes would include a set of changes that alter $C$ in such a way that:

$$C \not\sqsubseteq P$$

After changing $C$, one would again remove the axiom shown in (30) from the ontology. Unfortunately, this approach is not possible in $\mathcal{EL}$, because the logic lacks the necessary constructors to create such an inconsistency [128].

## 4.3. Computing the Difference between Concepts

This section focuses on work that aims at computing the difference between concepts. The employed methods are investigated and it is shown why they are not applicable to a fine-grained retraction of knowledge from $\mathcal{EL}$ concepts.

### 4.3.1. Concept Contraction

Colucci et al. [129] define a *concept contraction* problem that is geared towards matchmaking in electronic market places. It can be described as follows. For two concepts

$C$ and $P$, such that $\mathcal{T} \models C$, all splittings of $C$ into two concepts $G$ (for *give-up*) and $K$ (for *keep*) are searched such that $K$ and $P$ are both satisfiable in $\mathcal{T}$. More formally, all pairs $(G, K)$ are searched that have the following properties:

1. $C \equiv G \sqcap K$.
2. $K \sqcap P$ is satisfiable in $\mathcal{T}$.

From the result they then choose the splitting $(G', K')$ where $G'$ is most general w.r.t. concept subsumption. $K'$ resembles the part of $C$ that remains after the concept contraction. However, in $\mathcal{EL}$ the result would always be the trivial splitting $(\top, C)$. This is explained in the following. First, the splitting $(\top, C)$ has the aforementioned properties and is therefore always a valid splitting, independent of the actual content of $C$:

1. $C \equiv \top \sqcap C$.
2. $C \sqcap P$ is satisfiable in $\mathcal{T}$ for all concepts $C$ and $P$.[22]

Second, $(\top, C)$ is always the most general splitting because $\top$ subsumes all other concepts. The concept contraction by Colucci et al. [129] is therefore not offhandedly applicable to the Description Logic $\mathcal{EL}$.

#### 4.3.2. Other Difference Operations

Teege [130] defines a difference operation between two concepts. In the following, let this operation be denoted by $\ominus$ and let $A$ and $B$ be concepts such that:

$$B \sqsubseteq A$$

The operation is then defined as follows, where $\mathcal{L}$ denotes again the set of all $\mathcal{EL}$ concepts:

$$B \ominus A := \max_{\sqsubseteq}(\{C \in \mathcal{L} \colon A \sqcap C \equiv B\})$$

The result of this operation is the most general concept which in conjunction with $A$ again equals $B$. To give an example, consider the following concept definitions:

$$\begin{aligned} RedCar &\equiv Car \sqcap \exists hasPainting.RedPaint \\ RedMetallicCar &\equiv Car \sqcap \exists hasPainting.(RedPaint \sqcap \exists hasAdditive.MetalPigments) \end{aligned}$$

$RedCar$ resembles a car that has a red painting and $RedMetallicCar$ defines the concept of a car which has a polychromatic red-metallic painting. Such a paint is

[22] See Lemma 3 in Section 7.4.

obtained by adding metal flakes to the red paint. Now assume that one wants to remove the red paint from the car. The following operation gives the intended result:

$$\underbrace{\mathcal{T}^*(RedCar)}_{B} \ominus \underbrace{\exists hasPainting.RedPaint}_{A} \equiv \underbrace{Car}_{C}$$

This happens because the concept of a red car can be obtained again by adding back the concept $Car$ to the subtrahend:

$$\underbrace{\exists hasPainting.RedPaint}_{A} \sqcap \underbrace{Car}_{C} \equiv \underbrace{\mathcal{T}^*(RedCar)}_{B}$$

However, it is not possible to perform a fine-grained removal of concept parts. Consider the operation:

$$\mathcal{T}^*(RedMetallicCar) \ominus \exists hasPainting.\exists hasAdditive.MetalPigments$$

In this case one could expect to end up with a new concept that represents the same car as above, with the only change that the car now lacks the metallic look:

$$Car \sqcap \exists hasPainting.RedPaint$$

However, the operation does not change the concept at all:

$$\begin{aligned}&\underbrace{\mathcal{T}^*(RedMetallicCar)}_{B} \ominus \underbrace{\exists hasPainting.\exists hasAdditive.MetalPigments}_{A}\\ \equiv &\underbrace{\mathcal{T}^*(RedMetallicCar)}_{C}\end{aligned}$$

The reason for this is that there does not exist a concept $X$ such that:

1. $X$ is at least as general as $RedMetallicCar$:

$$\mathcal{T}^*(RedMetallicCar) \sqsubseteq X$$

2. $X$ semantically differs from $RedMetallicCar$:

$$X \not\equiv \mathcal{T}^*(RedMetallicCar)$$

3. Adding back the subtrahend to $X$ restores $RedMetallicCar$:[23]

$$\begin{aligned}&X \sqcap \exists hasPainting.\exists hasAdditive.MetalPigments\\ \equiv &Car \sqcap \exists hasPainting.(RedPaint \sqcap \exists hasAdditive.MetalPigments)\end{aligned}$$

In addition, as Suchanek et al. [131] points out, the operator is undefined in the case that $B \not\sqsubseteq A$.

Brandt et al. [132] propose a difference operator related to Teege's operator, but instead of choosing the most general concept w.r.t. concept subsumption they define an order based on the syntax of concepts. This operator behaves quite similar if the subtrahend subsumes the minuend and subsequently gives the same results in the previous two examples as Teege's.

[23] Note that $\exists R.A \sqcap \exists R.B \not\equiv \exists R.(A \sqcap B)$.

#### 4.3.3. The Subtraction Operator by Suchanek et al. [6]

Suchanek et al. [6, 131] propose a set of operators for the modification of $\mathcal{EL}$ concepts. In their work they explore how these operators can be used to generate new concepts in a creative way.[24] One of their operators is a subtraction operator which is denoted in the following by $\ominus$. It removes the first conjunct within a concept's definition that is subsumed by the subtrahend. To make the operator deterministic, they impose an order on concepts based on the lexicographical ordering of concept names and role names. Before the subtraction is performed, they reorder all conjuncts in the concept with respect to this order. Prior to this they simplify the concept and remove redundant parts such as multiple occurrences of the same conjunct. In the following, a more detailed explanation of this operator is given. This is then followed by an example that shows that the operator is not applicable to the task of a fine-grained removal of a consequence from a concept.

To compute $C \ominus P$, where $P$ is only a trivial conjunction,[25] first

$$C' = norm(C)$$

is obtained from $C$ by computing the so-called *normal form* of $C$.[26] This includes removing redundant conjuncts that occur within the definition of $C$, e.g.:

$$norm(A \sqcap \exists R.(B \sqcap \top) \sqcap A) = A \sqcap \exists R.B$$

Assume in the following w.l.o.g. that $norm(C)$ has the following form, where each $C'_i$ with $i \in \{1, 2, \ldots, n\}$ is a trivial conjunction:

$$norm(C) = C'_1 \sqcap C'_2 \sqcap \ldots \sqcap C'_n$$

In addition, let $\prec$ denote a total order between the concepts in $\mathcal{EL}$ that contain no redundant conjuncts. Assume that the conjuncts of $norm(C)$ have been syntactically rearranged such that:

$$\forall i \in \{1, 2, \ldots, n-1\} \colon C'_i \prec C'_{i+1}$$

The result of $C \ominus P$ is then given by the following case distinction:

**Case 1:** There exists at least one trivial conjunct in $norm(C)$ that is subsumed by $P$. Then it is possible to choose an index $k$ of a conjunct in $norm(C)$ such that:

$$C'_k \sqsubseteq P \textit{ and } \forall i \in \{1, 2, \ldots, k-1\} \colon C'_i \not\sqsubseteq P$$

[24] To determine the creativity of a new concept they conducted a small study where computer science students had to rate the created concepts.

[25] The case where $P$ is a conjunction of more than one concept is omitted here, because it is not needed in the following examples.

[26] For a formal definition of the normal form of a concept the reader is referred to Suchanek et al. [131].

The result of the operation is then defined as:

$$C \ominus P = norm((\prod_{i=1}^{k-1} C'_i) \sqcap (\prod_{i=k+1}^{n} C'_i))$$

In other words, the first conjunct that implies the subtrahend is removed from the conjunction.

**Case 2:** There does not exist a conjunct in $norm(C)$ that is subsumed by $P$. Then the result of the operation is defined as:

$$C \ominus P = C$$

It is now shown that this operator cannot be used for a fine-grained removal of conjuncts. Consider again the concept of a red car as it was defined in Section 4.3.2:

$$RedCar \equiv Car \sqcap \exists hasPainting.RedPaint$$

If one wants to get rid of the red color, the operator of Suchanek et al. [6] can be used to achieve this:

$$\mathcal{T}^*(RedCar) \ominus \exists hasPainting.RedPaint \equiv Car$$

However, there are situations in which the operator may not give the desired result. Similar to Teege's difference operation, the operator of Suchanek et al. [6] does not support a fine-grained removal of conjuncts. Consider again the concept of a red metallic car:

$$RedMetallicCar \equiv Car \sqcap \exists hasPainting.(RedPaint \sqcap \exists hasAdditive.MetalPigments)$$

Now assume again that one wants to remove the metallic look from the car. The intended result is again:

$$Car \sqcap \exists hasPainting.RedPaint$$

To achieve this, one could try to apply the following operation:

$$\mathcal{T}^*(RedMetallicCar) \ominus \exists hasPainting.\exists hasAdditive.MetalPigments$$

In contrast to Teege's, the operation returns a concept that differs from the minuend. But instead of returning the desired concept of a red car that lacks the metallic look, the operation returns the concept of a car that has no color at all:

$$\mathcal{T}^*(RedMetallicCar) \ominus \exists hasPainting.(\exists hasAdditive.MetalPigments) \equiv Car$$

This happens because the operator completely removes the first conjunct that is subsumed by $\exists hasPainting.(\exists hasAdditive.MetalPigments)$. In the example case, this is the painting of the car:

$$\exists hasPainting.(RedPaint \sqcap \exists hasAdditive.MetalPigments) \sqsubseteq \exists hasPainting.\exists hasAdditive.MetalPigments$$

The operator of Suchanek et al. [6] is therefore not applicable to the task of a fine-grained removal of knowledge from $\mathcal{EL}$ concepts.

## 4.4. Conclusion

This section examined related work. First, the postulates for belief set contraction and belief base contraction were analyzed and it was shown that the recovery and relevance postulates cannot naïvely be applied to a removal of knowledge at concept level. In a next step, an overview of research that aims at debugging and repairing ontologies was given. This overview included laconic justifications as well as other approaches that aim at resolving inconsistencies at a fine-grained level. Following this, methods that aim at computing the difference between concepts were examined. As a result, it was shown that these approaches cannot be used for a fine-grained removal of knowledge from concepts or cannot offhandedly be transferred to $\mathcal{EL}$.

The following chapter is therefore concerned with the development of an operator for the Description Logic $\mathcal{EL}$ that supports a fine-grained removal of knowledge from concepts. Such an operator would also be beneficial in combination with laconic justifications. These could be used to identify the part of a concept's definition that is responsible for a certain entailment. In a second step the operator could then be applied to the concept in order to retract this part.

# Part III.
# An Operator for Retracting Knowledge from $\mathcal{EL}$ Concepts

In this chapter, an operator for the fine-grained removal of knowledge from $\mathcal{EL}$ concepts is defined. For this, the normal form of a concept is introduced and an order between concepts is defined. Building on this, the requirements to the retraction operator are formalized. Here, the postulates for belief set contraction and belief base contraction are used as a guidance and related properties are derived. As a consequence from the findings of the previous section, additional properties are developed to cope with the non-applicability of the recovery postulate and the relevance postulate. After the fundamentals have been established a formal definition of the operator is given. Following this, the decisions in the design of the operator are explained and some example applications are shown. At the end of this section it is proven that the operator fulfills all the formalized requirements.

## 5. Preliminaries

This section introduces some basic definitions and lists assumptions that are required in the following.

### 5.1. Definitions

Let $\mathcal{L}$ denote the set of all $\mathcal{EL}$ concepts. Following Suchanek et al. [131], the full reduction of a concept and the normal form of a concept are defined as follows:

**Definition 15** (Full reduction of a concept)**.** *Let the reduction of a concept $C$ be a concept $C'$ such that:*

- $C' \sqsubseteq C$,
- *$C'$ does not introduce new conjuncts w.r.t. $C$,*
- *$C'$ has less conjuncts than $C$.*

*The full reduction $red(C)$ of a concept $C$ is then given by a concept $C'$ that has no reduction, i.e.:*

*$C'$ is the full reduction of $C$ iff there exists no reduction of $C'$*

Suchanek et al. [131] show that every concept that does not contain duplicate conjuncts has a unique full reduction. The normal form of a concept is given by the following definition, based again on Suchanek et al. [131].

**Definition 16** (Normalized concept)**.** *Let* $norm(C)$*, the* normal form *of a concept* $C$*, be defined as follows:*

$$norm(C) = \begin{cases} red(\prod_{i=1}^{n} norm(C_i)) & \text{if } C \text{ is a non-trivial conjunction of } n \text{ trivial conjuncts } C_i \text{ with } i \in \{1, 2, \ldots, n\} \\ \exists R.norm(C') & \text{if } C = \exists R.C' \text{ with a role } R \text{ and a concept } C' \\ C & \text{otherwise} \end{cases}$$

*In the following, a concept* $C$ *is called* normalized *iff* $C = norm(C)$*.*

**Example 8** (Normalized concept)**.** *Consider the following example results of the complete expansion of different concepts:*

$$\mathcal{T}^*(RedCar) = Car \sqcap \exists hasPart.Engine \sqcap \exists hasPainting.Red$$
$$\mathcal{T}^*(MotorVessel) = Boat \sqcap \exists hasPart.(Engine \sqcap \top)$$

*Only* $\mathcal{T}^*(RedCar)$ *is a normalized concept.* $\mathcal{T}^*(MotorVessel)$ *is not in normal form. This is shown in the following:*

$$\begin{aligned} & norm(\mathcal{T}^*(MotorVessel)) \\ \equiv\ & norm(Boat \sqcap \exists hasPart.(Engine \sqcap \top)) \\ \equiv\ & red(norm(Boat) \sqcap norm(\exists hasPart.(Engine \sqcap \top))) \\ \equiv\ & red(Boat \sqcap \exists hasPart.norm(Engine \sqcap \top)) \\ \equiv\ & red(Boat \sqcap \exists hasPart.red(norm(Engine) \sqcap norm(\top))) \\ \equiv\ & red(Boat \sqcap \exists hasPart.red(Engine \sqcap \top)) \\ \equiv\ & red(Boat \sqcap \exists hasPart.Engine) \\ \equiv\ & Boat \sqcap \exists hasPart.Engine \end{aligned}$$

As Suchanek et al. [131] point out, concepts that are equivalent and do not contain duplicate conjuncts have the same normal form. This is captured by the following Lemma:

**Lemma 1.** *Let* $C$ *and* $C'$ *be concepts. Then:*

$$\text{If } C \equiv C', \text{ then } norm(C) = norm(C')$$

For a proof of this reader is referred to Suchanek et al. [131]. In the following an order $\prec$ between normalized concepts is defined, based on the order given by Suchanek et al. [131].

**Definition 17** (Order)**.** *Let all concepts be a part of a completely expanded and normalized TBox. In addition, let all concepts be normalized.*[27] *Let the base symbols in* $\mathcal{B}_\mathcal{T} \backslash \{\top\}$ *be*

[27] Note again the difference between a normalized TBox (Definition 6 in Section 3.3.2) and the normalization of a concept.

*ordered by a complete order $\prec_{\mathcal{B}_\mathcal{T}}$ and let the role names in $\mathcal{R}_\mathcal{T}\backslash\{u\}$ be ordered by a complete order $\prec_{\mathcal{R}_\mathcal{T}}$. Let $A$ and $B$ denote concept names in $\mathcal{B}_\mathcal{T}\backslash\{\top\}$ and let $R$ and $S$ denote role names in $\mathcal{R}_\mathcal{T}\backslash\{u\}$. Let $C$ and $P$ denote arbitrary complex concepts. In addition, let $D, C_1, C_2, \ldots, C_n, P_1, P_2, \ldots, P_m$ denote concepts that are trivial conjunctions. The order $\prec$ is then defined as follows:*

$$
\begin{aligned}
C &\prec \top \quad \textit{for } C \neq \top \\
A &\prec B \quad \textit{for } A \prec_{\mathcal{B}_\mathcal{T}} B \\
\exists R.C &\prec A \\
\exists R.C &\prec \exists S.P \quad \textit{for } R \prec_{\mathcal{R}_\mathcal{T}} S \\
\exists R.C &\prec \exists R.P \quad \textit{for } C \prec P \\
\exists R.C &\prec \exists u.P \\
\exists u.C &\prec \exists u.P \quad \textit{for } C \prec P \\
(C_1 \sqcap C_2 \sqcap \ldots \sqcap C_n) &\prec D \quad \textit{for } n > 1 \textit{ and } C_1 \prec D \\
D &\prec (C_1 \sqcap C_2 \sqcap \ldots \sqcap C_n) \quad \textit{for } n > 1 \textit{ and } (D = C_1 \textit{ or } D \prec C_1) \\
(C_1 \sqcap C_2 \sqcap \ldots \sqcap C_n) &\prec (P_1 \sqcap P_2 \sqcap \ldots \sqcap P_m) \quad \textit{for } m > 1 \textit{ and } n > 1 \\
&\qquad \textit{and } (C_1 \prec P_1 \textit{ or } (C_1 = P_1 \textit{ and} \\
&\qquad (C_2 \sqcap \ldots \sqcap C_n) \prec (P_2 \sqcap \ldots \sqcap P_m)))
\end{aligned}
$$

*It is assumed that the conjuncts in every conjunction are already ordered by $\prec$.*

The introduction of the order $\prec$ makes it possible to define a lowest conjunct of a conjunction:

**Definition 18** (Lowest conjunct)**.** *Let $C = C_1 \sqcap C_2 \sqcap \ldots \sqcap C_n$ be a normalized concept, where each $C_i$ with $i \in \{1, 2, \ldots, n\}$ is a trivial conjunction. Then the* lowest conjunct *of $C$ is defined as the conjunct $C_i$ that is the smallest conjunct in $C$ w.r.t. the order $\prec$. More formally:*

$$
\begin{aligned}
&C_i \textit{ is the lowest conjunct of } C \textit{ iff } i \in \{1, 2, \ldots, n\} \textit{ and } \forall j \in \{1, 2, \ldots, n\}: \\
&\qquad\qquad \textit{if } i \neq j, \textit{ then } C_i \prec C_j
\end{aligned}
$$

*In addition, the lowest conjunct of a concept $C$ is denoted by $C_\prec$.*

The proofs that are later given in Section 7.4 are often based on an induction over the structure of a concept. To facilitate these proofs, the *parts of a concept* are now defined.

**Definition 19** (Parts of a concept)**.** *Let $C$ be a concept. Then the* parts of $C$ *are given by the following case distinction:*

***Case 1:*** *$C$ is a non-trivial conjunction of the form $C = C_1 \sqcap C_2 \sqcap \ldots \sqcap C_n$ such that each $C_i$ with $i \in \{1, 2, \ldots, n\}$ is a trivial conjunction. Then the conjuncts $C_1, C_2, \ldots, C_n$ are the parts of $C$.*

***Case 2:*** *$C$ is a role restriction of the form $C = \exists R.C_1$ with a role $R$ and a concept $C_1$. Then $C_1$ is the only concept that is a part of $C$.*

*In the case that $C$ is an atomic concept, the parts of $C$ are undefined.*

**Example 9** (Parts of a concept)**.** *Consider the following example definitions:*

$$C \equiv A \sqcap \exists R.B \tag{31}$$

$$C' \equiv \exists R.(A \sqcap B) \tag{32}$$

*In the case of* (31)*, the parts of $C$ are given by the concepts $A$ and $\exists R.B$. In the case of* (32)*, only the concept $A \sqcap B$ is a part of $C'$.*

## 5.2. Assumptions

This section lists some assumptions which are required for the definition of the knowledge retraction operator.

**Normalized TBox:** It is assumed that the TBox is normalized. If the TBox happens to be a generalized TBox it can be transformed into a normalized TBox as follows, based on Baader and Nutt [103]: For every concept inclusion axiom

$$C \sqsubseteq P$$

a new base symbol $\overline{C}$ is chosen. The axiom above is then replaced with the new axiom:

$$C \equiv \overline{C} \sqcap P$$

**Acyclic TBox:** It is assumed that the TBox is acyclic. Acyclic terminologies are still expressive enough for certain real-world ontologies such as SNOMED CT.

**Completely expanded concepts:** It is assumed that all concepts are completely expanded. Note that complete expansion is possible, because the TBox is required to be acyclic.

**No duplicate conjuncts:** No concept contains duplicate conjuncts. A concept $C$ that contains the same conjunct $P$ $m$ times can be trivially transformed into an equivalent concept $C'$ that contains no duplicate conjuncts by removing $m - 1$ occurrences of $P$ in $C$.

**Normalized concepts:** All concepts are normalized, i.e. for every concept $C$:

$$C = norm(C)$$

The normalization of all concepts is possible because no concept contains duplicate conjuncts. Note that the assumed normalization of all concepts together with the complete expansion of all concepts implies that a lowest conjunct exists for every concept $C$.

**TBox independent subsumption:** In the following, the semantics of concept relationships ignores the underlying TBox. Note that this does not lead to a change in the semantics, because all concepts are completely expanded [103]. More formal, given an acyclic TBox $\mathcal{T}$, a concept $C$ and a concept $P$ it is true that:

$$\mathcal{T} \models C \sqsubseteq P \textit{ iff } \models \mathcal{T}^*(C) \sqsubseteq \mathcal{T}^*(P)$$

For convenience, $\models$ is dropped if it is clear from the context that reasoning w.r.t. an empty TBox is meant.

**Exclusion of the universal role:** In the following, the universal role is not allowed to appear within the subtrahend of a contraction operation. Note that the universal role is still allowed to appear within the minuend and that this assumption therefore imposes no restriction on the content of the ontology.

In this section some basic assumptions were made. The next section outlines the requirements the operator must fulfill.

# 6. Outline

This section gives an overview of the requirements an $\mathcal{EL}$ contraction operator should have. First, the aim of the operator is defined. After that, the relationship between the operator and belief set and belief base contraction is analyzed. Based on this, properties are defined which the operator must fulfill and it is explained why the commutability of subtrahends conflicts with these properties.

## 6.1. Problem Description and Relationship to Belief Revision

The retraction operator has to support a fine-grained removal of consequences from $\mathcal{EL}$ concepts. An example for such a fine-grained removal is given in the following:

**Example 10** (Fine-grained removal)**.** *Let $\ominus$ denote the knowledge retraction operator. Then consider again the example definition of a car with a red metallic appearance:*

$$RedMetallicCar \equiv Car \sqcap \exists hasPainting.(RedPaint \sqcap \exists hasAdditive.MetalPigments)$$

*The operator should allow the removal of the metallic effect without removing the knowledge that the car is painted in red:*

$$\begin{aligned} &\mathcal{T}^*(RedMetallicCar) \ominus \exists hasPainting.\exists hasAdditive.MetalPigments \\ \equiv\ &Car \sqcap \exists hasPainting.RedPaint \end{aligned}$$

In addition, the operator should exhibit other properties such as a guaranteed success of the retraction operation. In the following section, the postulates for belief base and belief set contraction are used to infer these properties. Since the recovery and relevance postulates cannot offhandedly be used in this task, additional properties are developed to make up for this loss.[28]

[28] See again Section 4.1 for an explanation of this.

## 6.2. Requirements

This section lists desirable properties that an operator for retracting knowledge from $\mathcal{EL}$ concepts should have. First, properties are given that are derived from belief base contraction and belief set contraction. This is followed by a list of additional properties, such as properties that ensure a fine-grained removal of knowledge. At the end of this section it is shown why the commutability of subtrahends conflicts with these properties.

### 6.2.1. Properties Derived from Belief Contraction

The following properties are closely related to the AGM postulates for belief base and belief set contraction.

**Closedness:**

**Property 1** (Closedness)**.** *For concepts $C$ and $P$:*

$$C \ominus P \text{ is a concept}$$

This property ensures that the result of the knowledge retraction operation is again a concept. It is related to the closure postulate of belief base and belief set contraction.

**Inclusion:**

**Property 2** (Inclusion)**.** *For concepts $C$ and $P$:*

$$C \sqsubseteq C \ominus P$$

This property guarantees that no new knowledge is gained when knowledge is removed. To give an example, consider again the definition of a red car:

$$RedCar \equiv Car \sqcap \exists hasPainting.RedPaint \tag{33}$$

Now assume that one wants to remove the red paint from the car:

$$\mathcal{T}^*(RedCar) \ominus \exists hasPainting.RedPaint$$

The inclusion property restricts the possible outcomes of the operation. Without this property the operator could introduce new knowledge as shown in the following:

$$\mathcal{T}^*(RedCar) \ominus \exists hasPainting.RedPaint \equiv Car \sqcap \exists hasEquipment.SeatHeating$$

The property is related to the inclusion postulate of belief base contraction:

$$B - \phi \subseteq B$$

In this postulate, the result of the contraction operation appears left of the subset relationship whereas in the derived inclusion property, the result of the retraction operation subsumes the concept. This difference stems from the different semantics: if one removes some knowledge from the definition of an $\mathcal{EL}$ concept $C$ (e.g. the red paint from the red car), then the new concept $C'$ is more general than $C$ w.r.t. concept subsumption. Contrary to this, a belief set is a set of logical formulae. To give up a certain belief, one removes the corresponding formulae from the set and ends up with a subset of the previous belief base.

**Vacuity:**

**Property 3** (Vacuity)**.** *For concepts $C$ and $P$:*

$$\textit{If } C \not\sqsubseteq P\textit{, then } C \ominus P = C$$

The vacuity property ensures that a concept is only changed if it is subsumed by the subtrahend. Besides a change in the semantics of that concept this property prevents a change of its syntax. To give an example for the vacuity property, consider again the definition of a red car as shown in (33). Note that this definition does not imply a blue painting of the car, i.e.:

$$\mathcal{T}^*(RedCar) \not\sqsubseteq \exists hasPainting.BluePaint$$

From the vacuity property it follows that a removal of the blue paint is therefore not allowed to change the concept:

$$\mathcal{T}^*(RedCar) \ominus hasPainting.BluePaint = \mathcal{T}^*(RedCar)$$

The property is related to the corresponding property of belief base contraction:

$$\text{If } \phi \notin Cn(B), \text{ then } B - \phi = B$$

**Success:**

**Property 4** (Success)**.** *For concepts $C$ and $P$:*

$$\textit{If } \top \not\equiv P\textit{, then } C \ominus P \not\sqsubseteq P$$

This property ensures that a knowledge retraction operation is successful in that the result is not subsumed by the subtrahend. The top concept is excluded, because every concept is subsumed by $\top$ and thus a successful retraction of $\top$ is impossible. To give an example, consider again the definition of a red car in (33) and consider the following operation:

$$\mathcal{T}^*(RedCar) \ominus \exists hasPainting.RedPaint \tag{34}$$

Clearly:

$$\top \not\equiv \exists hasPainting.RedPaint$$

From the success property it therefore follows that the operation in (34) must remove the red painting from the car:

$$\mathcal{T}^*(RedCar) \ominus \exists hasPainting.RedPaint \not\sqsubseteq \exists hasPainting.RedPaint$$

It is related to the success postulate of belief base contraction:

$$\text{If } \phi \notin Cn(\emptyset), \text{ then } \phi \notin Cn(B - \phi)$$

Analogous to the exclusion of the top concept, $\phi$ is excluded from the postulate if it is a tautology and can therefore be inferred from an empty belief base.

**Preservation:**

**Property 5** (Preservation)**.** *For concepts $C$, $P$ and $P'$:*

$$\textit{If } P \equiv P', \textit{ then } C \ominus P \equiv C \ominus P'$$

The preservation property ensures that the retraction of semantically equivalent concepts gives the same results. It is related to the preservation postulate for belief set contraction:

$$\text{If } Cn(\phi) = Cn(\phi'), \text{ then } K - \phi = K - \phi'$$

### 6.2.2. Other Properties

There exist some other properties that are desirable for a knowledge retraction operator which have not been captured yet. This section discusses missing aspects of the retraction operation and presents corresponding properties.

**Destruction:** In the previous section, it was not defined how the operator should behave in the case that the subtrahend is equal to $\top$. To satisfy all the previous properties it would be sufficient to define the operator in such a way that the subtraction of $\top$ has no effect. However, it can be beneficial to let $\top$ work like a wildcard for every other concept $C$:

$$C \ominus \top \equiv \top \tag{35}$$

This is illustrated in the following. Consider again the definition of a red car:

$$RedCar \equiv Car \sqcap \exists hasPainting.RedPaint$$

If $\top$ worked like a placeholder, one could think of an operator that allows one to get rid of the color of the car without knowing its actual color:

$$\mathcal{T}^*(RedCar) \ominus \exists hasPainting.\top \not\sqsubseteq \exists hasPainting.RedPaint$$

Note that this does not imply that an operator with such a property cannot be used with $\top$ in a fine-grained way. The property in (35) does not conflict with the following result:

$$\mathcal{T}^*(RedCar) \ominus \exists hasPainting.\top \sqsubseteq Car$$

The idea to let $\top$ work like a placeholder for every other concept is captured by the *destruction* property, which is defined as follows:

**Property 6** (Destruction)**.** *For a concept $C$:*

$$C \ominus \top \equiv \top$$

**Minimality:** In the case of belief set contraction, the minimality of contraction follows from the recovery postulate and in the case of belief base contraction, it stems from the relevance postulate. As shown by Hansson [3], dropping the recovery postulate allows one to define a destructive AGM contraction operator:

$$K - \phi = \begin{cases} K \cap Cn(\emptyset) & \text{if } \phi \in Cn(K) \\ K & \text{otherwise} \end{cases}$$

Analogously, it is easy to see that the following example definition of an $\mathcal{EL}$ knowledge retraction operator would fulfill all the previously defined properties closedness, inclusion, vacuity, success, preservation and destruction:

$$C \ominus P = \begin{cases} \top & \text{if } C \sqsubseteq P \\ C & \text{otherwise} \end{cases}$$

It is therefore necessary to define some additional properties that ensure that the operator acts in a less destructive way. However, despite the fact that the recovery postulate and the relevance postulate cannot be naïvely applied, the idea behind these postulates can still be put into use. Consider again the relevance postulate as shown in Section 4.1. $B$ denotes again a belief base and $Cn$ denotes the consequence relation:

*If $\beta \in B$ and $\beta \notin B - \phi$, then there exists a set $B'$ such that $B - \phi \subseteq B' \subseteq B$ and $\phi \notin Cn(B')$, but $\phi \in Cn(B' \cup \{\beta\})$.*

The idea behind this postulate is that if some belief $\beta$ is removed during the contraction of a belief $\phi$, then $\beta$ must have played some role in the previous entailment of $\phi$. In the following, this idea is exploited to define a property that enforces a conservative removal of knowledge. For this, the term *independent concepts* is defined first.

**Definition 20** (Independent concepts)**.** *Let $C = \bigsqcap_{i=1}^{n} C_i$ and $P = \bigsqcap_{i=1}^{m} P_i$ be concepts, such that $C_i$ and $P_j$ are trivial conjunctions with $i \in \{1, 2, \ldots, n\}$ and $j \in \{1, 2, \ldots, n\}$. Then $C$ and $P$ are independent iff $\forall i \in \{1, 2, \ldots, n\}$ and $\forall j \in \{1, 2, \ldots, m\}$: $C_i \not\sqsubseteq P_j$ and $P_j \not\sqsubseteq C_i$.*

**Example 11** (Independent concepts)**.** *Consider the concept definitions of a red car, a red metallic car and a blue plane:*

$$\begin{aligned} RedCar &\equiv Car \sqcap \exists hasPainting.RedPaint \\ RedMetallicCar &\equiv Car \sqcap \exists hasPainting.(RedPaint \\ &\quad \sqcap \exists hasAdditive.MetalPigments) \\ BluePlane &\equiv Plane \sqcap \exists hasPainting.BluePaint \end{aligned}$$

*Assume now that these axioms are the content of a TBox.* $\mathcal{T}^*(BluePlane)$ *is then independent from* $\mathcal{T}^*(RedCar)$*, because no conjuncts within their definitions subsume each other.* $\mathcal{T}^*(RedCar)$ *and* $\mathcal{T}^*(RedMetallicCar)$ *are not independent, because they contain conjuncts that subsume each other:*

$$\begin{aligned} &\exists hasPainting.(RedPaint \sqcap \exists hasAdditive.MetalPigments) \\ &\sqsubseteq \exists hasPainting.RedPaint \end{aligned}$$

By making use of this definition, the first property that enforces a conservative removal of knowledge is defined as follows:

**Property 7** (Independence)**.** *For concepts* $C$*,* $P$ *and* $P'$*:*

*If* $C \sqsubseteq P'$ *and* $P$ *and* $P'$ *are independent, then* $C \ominus P \sqsubseteq P'$

Independence guarantees that conjuncts that are not related to the subtrahend are not removed. To give an example, consider again the definition of a red car from Example 11:

$$RedCar \equiv Car \sqcap \exists hasPainting.RedPaint$$

Then consider the following operation:

$$\mathcal{T}^*(RedCar) \ominus \exists hasPainting.RedPaint$$

Every operator that fulfills the independence property must keep the conjunct $Car$ in the result, because it is implied by $\mathcal{T}^*(RedCar)$ and independent from the subtrahend. However, even with this property in place there is still a case left that must be considered. Until now, it is not defined how the operator should behave if the subtrahend is a non-trivial conjunction of concepts, such as:

$$(A \sqcap B) \ominus (A \sqcap B) \tag{36}$$

The independence property does not apply here, because the concepts within the subtrahend are clearly not independent from the concepts that occur left-hand. Two different approaches are possible in this case. First, one could

define the operator in such a way that every conjunct that appears within the subtrahend is removed by an own retraction operation. This is shown in the following example definition where role restrictions are excluded for reasons of simplification.

**Definition 21** (Example definition of $\ominus$)**.** *Let $C = \bigsqcap_{i=1}^{n} C_i$ and $P = \bigsqcap_{j=1}^{m} P_j$ be concepts where $C_i$ and $P_j$ with $i \in \{1, 2, \ldots, n\}$ and $j \in \{1, 2, \ldots, m\}$ are trivial conjunctions. Then:*

$$C \ominus P = \begin{cases} \top & \text{$C$ and $P$ are a trivial conjunctions and } C \sqsubseteq P \\ C \ominus P_1 \ominus P_2 \ominus \ldots \ominus P_m & \text{$P$ is a non-trivial conjunction and } C \sqsubseteq P \\ (C_1 \ominus P) \sqcap \ldots \sqcap (C_n \ominus P) & \text{$C$ is a non-trivial conjunction and $P$ is a trivial conjunction and } C \sqsubseteq P \\ C & \text{otherwise} \end{cases}$$

*In addition, let $\ominus$ be left-associative.*

Using this definition, the example in (36) would result in:

$$\begin{aligned} & (A \sqcap B) \ominus (A \sqcap B) \\ = {} & (A \sqcap B) \ominus A \ominus B \\ = {} & ((A \sqcap B) \ominus A) \ominus B \\ = {} & ((A \ominus A) \sqcap (B \ominus A)) \ominus B \\ = {} & (\top \sqcap B) \ominus B \\ \equiv {} & B \ominus B \\ = {} & \top \end{aligned}$$

This definition would have the drawback that a retraction of $A \sqcap B$ would have more impact than retracting one of the more general concepts $A$ or $B$, which would however have less impact than retracting the even more general concept $\top$. This is shown in the following:

$$(A \sqcap B) \ominus (A \sqcap B) \equiv \top \tag{37}$$

$$(A \sqcap B) \ominus A \equiv B \tag{38}$$

$$(A \sqcap B) \ominus \top \equiv \top \tag{39}$$

For this reason another approach was considered. First it was noted that from the normalization of all concepts it follows that to fulfill all the aforementioned properties, especially success, it is sufficient to remove only one of the conjuncts which appear within the subtrahend.[29] In other words, for every

[29] This is indirectly proven in Section 7.4.

normalized concept $\bigsqcap_{i=1}^{n} C_i$ where each $C_i$ with $i \in \{1, 2, \ldots, n\}$ is a trivial conjunction:

$$\forall i \in \{1, 2, \ldots, n\} \colon C_1 \sqcap C_2 \sqcap \ldots \sqcap C_{i-1} \sqcap C_{i+1} \sqcap C_{i+2} \sqcap \ldots \sqcap C_n \not\sqsubseteq C_1 \sqcap C_2 \sqcap \ldots \sqcap C_n$$

In the following, a property called *informational economy* is defined that makes use of this finding. It is then shown that this property prevents results such as shown in (37) – (39).

**Property 8** (Informational economy)**.** *For concepts $C$ and $P$:*

$$\textit{If } C \sqsubseteq P\textit{, then } C \ominus P = C \ominus P_{\prec}$$

This property guarantees that if a conjunction of concepts is retracted not all concepts are removed but only enough to guarantee success of the operation. In addition, it ensures that an order $\prec$ between concepts is taken into account. Only the concept that is lowest w.r.t. this order is retracted. This property is motivated by the *principle of informational economy* [133]. This principle states that unnecessary losses of information should be avoided and reflects the underlying motivation in the design of the AGM postulates for belief set contraction. Consider the following modified version of Definition 21 that follows the informational economy property:[30]

**Definition 22** (Second example definition of $\ominus$)**.** *Let $C = \bigsqcap_{i=1}^{n} C_i$ and $P = \bigsqcap_{j=1}^{m} P_j$ be concepts, where each $C_i$ and $P_j$ are trivial conjunctions. Then:*

$$C \ominus P = \begin{cases} \top & \textit{$C$ and $P$ are a trivial conjunctions and } C \sqsubseteq P \\ C \ominus P_{\prec} & \textit{$P$ is a non-trivial conjunction and } C \sqsubseteq P \\ (C_1 \ominus P) \sqcap \ldots \sqcap (C_n \ominus P) & \textit{$C$ is a non-trivial conjunction and $P$ is a trivial conjunction and } C \sqsubseteq P \\ C & \textit{otherwise} \end{cases}$$

*In addition, let $\ominus$ be left-associative.*

It is now shown that this property prevents the results shown in (37) – (39). For this, let all concept names be ordered alphabetically, i.e. $A \prec B$. Then:

$$\begin{aligned} (A \sqcap B) \ominus (A \sqcap B) &= (A \sqcap B) \ominus A \equiv B \\ &(A \sqcap B) \ominus A \equiv B \\ &(A \sqcap B) \ominus \top \equiv \top \end{aligned}$$

[30] Note that role restrictions are omitted again for reasons of simplification.

Here, the retraction of a more general concept has as least as much impact as the retraction of a less general concept. In addition, this property makes the definition of favored retraction candidates possible, based e.g. on the epistemic entrenchment of certain concepts.[31] This is related to the computation of a *reject set* in Fuhrmann [134] where the author defines a *minimal theory contraction* and a *minimal base contraction*. The sentences that are retracted are chosen from a possible larger set of rejection candidates, based on an order between them that reflects their so-called *degree of retractability*.

This section introduced some additional properties that are considered important for a knowledge retraction operator and gave an explanation why they were chosen. At first, the commutability of subtrahends was considered as an additional property, but this idea was later discarded. The next section addresses this decision and explains the reasons for it.

### 6.2.3. Dismissed Property: Commutability

The commutability of subtrahends was considered as an additional property for the operator. However, it was dismissed when it turned out that this property conflicts with other properties that were valued more important. In the following, a definition for a property is given that expresses commutability of subtrahends. After this, it is proven that this property conflicts with some of the properties that were introduced in the previous sections.

**Property 9** (Commutability)**.** *For concepts $C$, $P_1$ and $P_2$:*

$$(C \ominus P_1) \ominus P_2 \equiv (C \ominus P_2) \ominus P_1$$

It is now shown that this property conflicts with the properties closedness, inclusion, vacuity, success and independence. For this, the proof first assumes that commutability complies with these properties. During the proof it is then shown that this assumption leads to a logical contradiction. To improve the readability of the proof, the corresponding placeholder concepts from the definition of the properties are displayed below the equation if one of these properties is used to infer a result. To give an example, if $A$, $B$ and $C$ were concepts, the commutability property (Property 9) could be used to derive the following equivalence:

$$(A \ominus B) \ominus C \equiv (A \ominus C) \ominus B$$

In the proof this is then written as:

$$(\underbrace{A}_{C} \ominus \underbrace{B}_{P_1}) \ominus \underbrace{C}_{P_2} \equiv (\underbrace{A}_{C} \ominus \underbrace{C}_{P_2}) \ominus \underbrace{B}_{P_1}$$

[31] An example for this is given with Example 15 in Section 7.3.

*Violation of Property 9.* Let $\ominus$ denote an operator that follows closedness (Property 1), inclusion (Property 2), vacuity (Property 3), success (Property 4), independence (Property 7) and commutability (Property 9). Let $P$ be a concept and let $A$ and $B$ be atomic concepts. In addition, let

$$A \not\equiv B$$

and let

$$A \sqcap B \sqsubseteq P$$

Moreover, let all concepts be normalized. Then consider the following operation:

$$(A \sqcap B) \ominus (A \sqcap B)$$

Depending on the actual definition of the operator different outcomes are possible. This is handled by the following two cases:

**Case 1:** $(A \sqcap B) \ominus (A \sqcap B) \equiv \top$. Then from this it follows that:

$$((A \sqcap B) \ominus (A \sqcap B)) \ominus P \equiv \top \ominus P \tag{40}$$

From the inclusion property (Property 2) it follows that:

$$\underbrace{\top}_{C} \sqsubseteq \underbrace{\top}_{C} \ominus \underbrace{P}_{P}$$

From this it is implied that:

$$\top \ominus P \equiv \top$$

From this and (40) it can be concluded that:

$$((A \sqcap B) \ominus (A \sqcap B)) \ominus P \equiv \top \ominus P \equiv \top \tag{41}$$

From the success property (Property 4) it follows that:

$$\underbrace{(A \sqcap B)}_{C} \ominus \underbrace{A}_{P} \not\sqsubseteq \underbrace{A}_{P} \tag{42}$$

Now let:

$$P = A$$

From this and (42) it is implied that:

$$(A \sqcap B) \ominus P \not\sqsubseteq A$$

From this, the semantics of $\mathcal{EL}$ and set theory it follows that:

$$(A \sqcap B) \ominus P \not\sqsubseteq A \sqcap B$$

From this and the vacuity property (Property 3) it follows that:

$$\underbrace{((A \sqcap B) \ominus P)}_{C} \ominus \underbrace{(A \sqcap B)}_{P} = \underbrace{(A \sqcap B) \ominus P}_{C} \tag{43}$$

Clearly:

$$A \sqcap B \sqsubseteq B$$

From this, the fact that $A$ and $B$ are independent concepts (Definition 20) and the independence property (Property 7) it follows that:

$$\underbrace{(A \sqcap B)}_{C} \ominus \underbrace{A}_{P} \sqsubseteq \underbrace{B}_{P'}$$

From this and $P = A$ it follows that:

$$(A \sqcap B) \ominus P \sqsubseteq B$$

From this and (43) it is implied that:

$$\underbrace{((A \sqcap B) \ominus P) \ominus (A \sqcap B)}_{\equiv (A \sqcap B) \ominus P} \sqsubseteq B \tag{44}$$

Note that the formula below the bracket shows the equivalence from (43). From the commutability property (Property 9) it now follows that:

$$(\underbrace{(A \sqcap B)}_{C} \ominus \underbrace{(A \sqcap B)}_{P_1}) \ominus \underbrace{P}_{P_2} \equiv (\underbrace{(A \sqcap B)}_{C} \ominus \underbrace{P}_{P_2}) \ominus \underbrace{(A \sqcap B)}_{P_1} \tag{45}$$

Note that the right side of (45) and the left side of (44) are equivalent. From this it can therefore be concluded that:

$$((A \sqcap B) \ominus (A \sqcap B)) \ominus P \sqsubseteq B$$

This conflicts with (41). In this case, commutability of subtrahends is therefore not possible if the other properties are kept.

**Case 2:** $(A \sqcap B) \ominus (A \sqcap B) \not\equiv \top$. Then from the inclusion property (Property 2) it follows that:

$$\underbrace{(A \sqcap B)}_{C} \sqsubseteq \underbrace{(A \sqcap B)}_{C} \ominus \underbrace{(A \sqcap B)}_{P}$$

From this and the normalization of all concepts it follows that the result from

$$(A \sqcap B) \ominus (A \sqcap B) \tag{46}$$

contains no other conjuncts than $A$ and $B$. From the success property (Property 4) it follows that:

$$\underbrace{(A \sqcap B)}_{C} \ominus \underbrace{(A \sqcap B)}_{P} \not\sqsubseteq \underbrace{A \sqcap B}_{P}$$

This implies that the result of (46) cannot contain both $A$ and $B$ as conjuncts. This only leaves $A$ or $B$ as a result. Assume w.l.o.g. that:

$$(A \sqcap B) \ominus (A \sqcap B) = A \tag{47}$$

Choose $P$ equal to this result:

$$P = A$$

Note that from this and (47) it follows that:

$$(A \sqcap B) \ominus (A \sqcap B) \sqsubseteq P$$

Note also that from (47) it is implied that:

$$(A \sqcap B) \ominus (A \sqcap B) \not\sqsubseteq B$$

From (47) and $P = A$ it follows that:

$$((A \sqcap B) \ominus (A \sqcap B)) \ominus P = A \ominus P = A \ominus A \tag{48}$$

From $P = A$ and the success property (Property 4) it follows that:

$$\underbrace{(A \sqcap B)}_{C} \ominus \underbrace{P}_{P} \not\sqsubseteq \underbrace{A}_{P}$$

From this, the semantics of $\mathcal{EL}$ and set theory it is implied that:

$$(A \sqcap B) \ominus P \not\sqsubseteq A \sqcap B$$

Clearly:

$$A \sqcap B \sqsubseteq B$$

From this, the fact that $A$ and $B$ are independent concepts (Definition 20) and the independence property (Property 7) it follows that:

$$\underbrace{(A \sqcap B)}_{C} \ominus \underbrace{A}_{P} \sqsubseteq \underbrace{B}_{P'} \tag{49}$$

From the success property (Property 4) it follows that:

$$A \ominus A \not\sqsubseteq A$$

From the inclusion property (Property 2) it is implied that:

$$\underbrace{A}_{C} \sqsubseteq \underbrace{A}_{C} \ominus \underbrace{A}_{P} \tag{50}$$

Clearly:

$$A \not\sqsubseteq B$$

From this, (50) and set theory it follows that:

$$A \ominus A \not\sqsubseteq B \tag{51}$$

From the success property (Property 4) it can be concluded that:

$$\underbrace{(A \sqcap B)}_{C} \ominus \underbrace{A}_{P} \not\sqsubseteq \underbrace{A}_{P}$$

From this and set theory it follows that:

$$(A \sqcap B) \ominus A \not\sqsubseteq A \sqcap B$$

From this and the vacuity property (Property 3) it is implied that:

$$\underbrace{((A \sqcap B) \ominus A)}_{C} \ominus \underbrace{(A \sqcap B)}_{P} = \underbrace{(A \sqcap B) \ominus A}_{C} \tag{52}$$

From the commutability property (Property 9) it again follows that:

$$(\underbrace{(A \sqcap B)}_{C} \ominus \underbrace{(A \sqcap B)}_{P_1}) \ominus \underbrace{P}_{P_2} \equiv (\underbrace{(A \sqcap B)}_{C} \ominus \underbrace{P}_{P_2}) \ominus \underbrace{(A \sqcap B)}_{P_1} \tag{53}$$

Note that from $P = A$ it follows that the left side of the equivalence in (52) equals the right side of the equivalence in (53). It can therefore be concluded that:

$$((A \sqcap B) \ominus (A \sqcap B)) \ominus P \equiv (A \sqcap B) \ominus A \tag{54}$$

The left side of (48) is equivalent to the left side of (54). It therefore follows that:

$$A \ominus A \equiv (A \sqcap B) \ominus A$$

Together with (51) it can be concluded that the left side of this equation is not subsumed by $B$. However, from (49) it follows that the right side is subsumed by $B$. This is a contradiction. The commutability of subtrahends is therefore not possible if the other properties are kept.

□

### 6.3. Conclusion

In this section the requirements to an $\mathcal{EL}$ knowledge retraction operator were stated. For this, the general aim of such an operator was defined. After that, the relationship to belief set contraction and belief base contraction was analyzed and desirable properties of the operator were defined. It was further shown why the commutability of subtrahends conflicts with these properties.

In the next section, the findings from this section are used to define an operator for the fine-grained retraction of knowledge from $\mathcal{EL}$ concepts.

## 7. Realization

In this section, an operator for the retraction of knowledge from $\mathcal{EL}$ concepts is defined with respect to the assumptions that were made in Section 5.2. The operator is based on the requirements that were given in the previous section. This includes the properties closedness (Property 1), inclusion (Property 2), vacuity (Property 3), success (Property 4), preservation (Property 5), destruction (Property 6), independence (Property 7) and informational economy (Property 8). After this, the decisions in the design of the operator are explained and some examples are given that show that the operator behaves as intended. In addition, formal proofs are provided that show that the operator fulfills the aforementioned properties.

### 7.1. Definition

Prior to the definition of the retraction operator $\ominus$, the auxiliary function $f_{contract}$ is defined which is used to facilitate the formal proofs.[32]

**Definition 23** (Contraction function $f_{contract}$)**.** *Let $\mathcal{L}$ denote the set of all $\mathcal{EL}$ concepts. Then $f_{contract}$: $\mathcal{L} \to \mathcal{L}$ is defined as follows:*

$$f_{contract}(C) = \begin{cases} \top & \text{if there exists } R \in \mathcal{R}_\mathcal{T} \text{ such that } C = \exists R.C' \text{ and } C' \equiv \top \\ C & \text{otherwise} \end{cases}$$

If this function is applied to a role restriction whose filler is equivalent to the top concept, then it replaces the whole role restriction with $\top$. Otherwise the argument is returned unchanged. Building upon this function, the retraction operator $\ominus$ is defined as follows.

**Definition 24** (Retraction operator $\ominus$)**.** *Let $C$, $P$ and $P'$ be concepts and let $P_\prec$ denote the lowest conjunct of $P$. The result of $C \ominus P$ is given by the following cases:*[33]

[32] This is discussed more in-detail in Section 7.2. In addition, Section 7.2 gives an alternate definition of the operator without an auxiliary function.

[33] Note that from the assumptions made in Section 5.2 it follows that the universal role is not allowed to occur within $P$.

***Case 1:*** *$C$ is an atomic concept. Then:*

$$C \ominus P = \begin{cases} \top & \text{if } C \sqsubseteq P \\ C & \text{otherwise} \end{cases}$$

***Case 2:*** *$C$ is a non-trivial conjunction of concepts $C_i$ such that $C = C_1 \sqcap C_2 \sqcap \ldots \sqcap C_n$ and every $C_i$ is a trivial conjunction. Then:*

$$C \ominus P = \begin{cases} \bigsqcap_{i=1}^{n} (C_i \ominus P_{\prec}) & \text{if } C \sqsubseteq P \\ C & \text{otherwise} \end{cases}$$

***Case 3:*** *$C$ is a role restriction such that $C = \exists R.C_1$, with concepts $C_1$ and $P_1$ and $R \in \mathcal{R}_{\mathcal{T}}$. Then:*[34]

$$C \ominus P = \begin{cases} f_{contract}(\exists R.(C_1 \ominus P_1)) & \text{if } C \sqsubseteq P \text{ and } P_{\prec} = \exists R.P_1 \\ \top & \text{if } P \equiv \top \\ C & \text{otherwise} \end{cases}$$

*In addition, the operator is defined to be left-associative:*

$$C \ominus P \ominus P' = (C \ominus P) \ominus P'$$

An explanation of the decisions that were taken in the design of the operator is given in the next section. This includes an explanation why role restrictions are removed if the filler of a role equals $\top$.

## 7.2. Remarks and Design Decisions

This section explains the choices that were taken in the design of the operator and gives some additional remarks. First note that the operator is well-defined:

**Lemma 2** (Well-definedness of $\ominus$)**.** *$C \ominus P$ is well-defined for all concepts $C$ and $P$ with $u \notin Roles(P)$.*

*Proof.* Well-definedness of $\ominus$ follows immediately from its definition. The only case that is less obvious is case 3. Here, $P \equiv \top$ and $P_{\prec} = \exists R.P_1$ are mutual exclusive, because the universal role is not allowed to appear in the subtrahend.[35] □

The removal of role restrictions posed a decision in the design of the operator. The need to remove role restrictions follows from the success property (Property 4). Consider the following example:

$$\exists R.\top \ominus \exists R.\top$$

[34] Note that if $P \not\equiv \top$ and $C \sqsubseteq P$, then it always follows that $P_{\prec}$ is a role restriction with the role $R$. This is later proven in Lemma 10 (Section 7.4).

[35] See also Lemma 10 (Section 7.4) and the definition of the order $\prec$ (Definition 17).

Here, from the success property it follows that:

$$(\exists R.\top \ominus \exists R.\top) \not\sqsubseteq \exists R.\top$$

The filler of $R$ cannot be changed, because it already equals $\top$. To cope with this, the decision was made that in this case the whole role restriction is replaced with $\top$. In addition, since the idea behind the allowance of $\top$ in the subtrahend was to provide a placeholder for an arbitrary concept, it was decided that a retraction of $\exists R.\top$ should not semantically differ from the retraction of some other concept $\exists R.C$. This led to the decision that if the filler of a role restriction is changed to $\top$, the whole role restriction is replaced with $\top$.

The function $f_{contract}$ prevents some unfortunate occurrences of the operator in its own definition and therefore facilitates the formal proofs. It is however not needed, as shown in the following:

**Definition 25** (Alternate definition of $\ominus$)**.** *Let $\ominus$ be defined as in Definition 24 with the following modification: In the case that $C$ is a role restriction such that $C = \exists R.C_1$, with concepts $C_1$ and $P_1$ and $R \in \mathcal{R}_\mathcal{T}$, let $C \ominus P$ instead be defined as follows:*

$$C \ominus P = \begin{cases} \exists R.(C_1 \ominus P_1) & \text{if } C \sqsubseteq P \text{ and } P_\prec = \exists R.P_1 \text{ and } C_1 \ominus P_1 \not\equiv \top \\ \top & \text{if } C \sqsubseteq P \text{ and } (P \equiv \top \text{ or } (P_\prec = \exists R.P_1 \text{ and } C_1 \ominus P_1 \equiv \top)) \\ C & \text{otherwise} \end{cases}$$

This case distinction is logically equivalent to case 3 in the definition of the operator. Note that in this definition the operator occurs on the right-hand side of the case distinction.

In addition, the operator was defined to be left-associative to prevent counter-intuitive results. A right-associative contraction operator would behave in a destructive way if an atomic concept is contracted multiple times. The following example shows this:

**Example 12** (Shortcomings of a right-associative operator)**.** *Consider again the definition of a red car:*

$$RedCar \equiv Car \sqcap \exists hasPainting.RedPaint$$

*If one would perform a multiple retraction of the concept $Car$ from this concept, one would expect to end up with a concept that still describe a class of things that are painted in red. A right-associative operator would instead give the following result:*

$$\begin{aligned} & \mathcal{T}^*(RedCar) \ominus Car \ominus Car \\ = \; & \mathcal{T}^*(RedCar) \ominus (Car \ominus Car) \\ \equiv \; & \mathcal{T}^*(RedCar) \ominus \top \\ \equiv \; & \top \end{aligned}$$

To further illustrate the definition of $\ominus$, some example applications of the operator are given in the following section. This is followed by formal proofs that show that the operator fulfills all the properties that were defined in Section 6.2.

### 7.3. Examples

This section presents some example applications of the operator. The first example shows how the operator can be used for a fine-grained removal of consequences from concepts. In the second example it is shown how conjunctions in the subtrahend are handled. Note that all concepts are normalized.[36]

**Example 13** (Fine-grained removal)**.** *In the following, the contraction function $f_{contract}$ is abbreviated by $f$. Consider again the definition of a car with a red-metallic appearance:*

$$\begin{aligned} RedMetallicCar \equiv\ & Car \sqcap \exists hasPainting.(RedPaint \\ & \sqcap \exists hasAdditive.MetalPigments) \end{aligned}$$

*It is now shown that the operator $\ominus$ can retract the knowledge that the car's paint has a metallic effect without removing the knowledge that the car is painted in red. First, the retraction operation is applied to the completely expanded concept $RedMetallicCar$:*

$$\begin{aligned} & \mathcal{T}^*(RedMetallicCar) \ominus \exists hasPainting.\exists hasAdditive.MetalPigments \\ \equiv\ & (Car \sqcap \exists hasPainting.(RedPaint \sqcap \exists hasAdditive.MetalPigments)) \\ & \ominus \exists hasPainting.\exists hasAdditive.MetalPigments \end{aligned}$$

*Since the minuend is a non-trivial conjunction case 2 of the definition of $\ominus$ is applied leading to:*

$$\begin{aligned} & (Car \sqcap \exists hasPainting.(RedPaint \sqcap \exists hasAdditive.MetalPigments)) \\ & \ominus \exists hasPainting.\exists hasAdditive.MetalPigments \\ \equiv\ & (Car \ominus \exists hasPainting.\exists hasAdditive.MetalPigments) \\ & \sqcap (\exists hasPainting.(RedPaint \sqcap \exists hasAdditive.MetalPigments) \\ & \ominus \exists hasPainting.\exists hasAdditive.MetalPigments) \end{aligned}$$

*Clearly:*

$$Car \not\sqsubseteq \exists hasPainting.\exists hasAdditive.MetalPigments$$

*The first conjunct is therefore not changed. The application of $\ominus$ to the second conjunct leads to case 3, where the minuend is a role restriction. Note that the subtrahend subsumes the minuend:*

$$\begin{aligned} & \exists hasPainting.(RedPaint \sqcap \exists hasAdditive.MetalPigments) \\ & \sqsubseteq \exists hasPainting.\exists hasAdditive.MetalPigments \end{aligned}$$

*The operation is therefore pushed inside the filler of the role restriction. In addition, the*

[36] See Section 5 again for details on this.

*outer role restriction in the subtrahend is dropped and the contraction function is applied:*

$$\begin{aligned}
&(Car \ominus \exists hasPainting.\exists hasAdditive.MetalPigments)\\
&\sqcap (\exists hasPainting.(RedPaint \sqcap \exists hasAdditive.MetalPigments)\\
&\ominus \exists hasPainting.\exists hasAdditive.MetalPigments)\\
\equiv\ &Car \sqcap f(\exists hasPainting.((RedPaint \sqcap \exists hasAdditive.MetalPigments)\\
&\ominus \exists hasAdditive.MetalPigments))
\end{aligned}$$

*Since the minuend is a non-trivial conjunction, case 2 is applied again, leading to:*

$$\begin{aligned}
&Car \sqcap f(\exists hasPainting.((RedPaint \sqcap \exists hasAdditive.MetalPigments)\\
&\ominus \exists hasAdditive.MetalPigments))\\
\equiv\ &Car \sqcap f(\exists hasPainting.((RedPaint \ominus \exists hasAdditive.MetalPigments)\\
&\sqcap (\exists hasAdditive.MetalPigments \ominus \exists hasAdditive.MetalPigments)))
\end{aligned}$$

*Since $RedPaint$ is an atomic concept, case 1 is applied. It is not changed, because it is not subsumed by $\exists hasAdditive.MetalPigments$. The second application of $\ominus$ leads again to case 3, and the operation is pushed inside the filler of $\exists hasAdditive$:*

$$\begin{aligned}
&Car \sqcap f(\exists hasPainting.((RedPaint \ominus \exists hasAdditive.MetalPigments)\\
&\sqcap (\exists hasAdditive.MetalPigments \ominus \exists hasAdditive.MetalPigments)))\\
\equiv\ &Car \sqcap f(\exists hasPainting.(RedPaint \sqcap f(\exists hasAdditive.(MetalPigments\\
&\ominus MetalPigments))))
\end{aligned}$$

*$MetalPigments$ is changed to $\top$, because it is an atomic concept and is subsumed by the subtrahend. This leads to:*

$$\begin{aligned}
&Car \sqcap f(\exists hasPainting.(RedPaint \sqcap f(\exists hasAdditive.(MetalPigments\\
&\ominus MetalPigments))))\\
\equiv\ &Car \sqcap f(\exists hasPainting.(RedPaint \sqcap f(\exists hasAdditive.\top)))
\end{aligned}$$

*The filler of the role restriction $\exists hasAdditive.\top$ equals $\top$, and thus $f_{contract}$ changes the whole concept to $\top$:*

$$\begin{aligned}
&Car \sqcap f(\exists hasPainting.(RedPaint \sqcap f(\exists hasAdditive.\top)))\\
\equiv\ &Car \sqcap f(\exists hasPainting.(RedPaint \sqcap \top))\\
\equiv\ &Car \sqcap f(\exists hasPainting.RedPaint)
\end{aligned}$$

*The filler of the role restriction differs from $\top$, and thus the outer application of $f_{contract}$ does not change the concept. The result therefore is given by:*

$$\begin{aligned}
&Car \sqcap f(\exists hasPainting.RedPaint)\\
\equiv\ &Car \sqcap \exists hasPainting.RedPaint
\end{aligned}$$

*This is the intended result which shows that the operator can be used for a fine-grained removal of consequences.*

The next example shows how the operator can be combined with laconic justifications which were introduced in Section 4.2.1.

**Example 14** (Laconic justifications)**.** *Consider the following generalized TBox:*

$$\begin{aligned} RedMetallicCar &\equiv Car \sqcap \exists hasPainting.(RedPaint \sqcap \exists hasAdditive.MetalPigments) \\ RedPaint &\sqsubseteq Paint \\ HardRepairablePaint &\equiv Paint \sqcap \exists hasAdditive.MetalPigments \end{aligned}$$

*This TBox differs from the previous example in that it contains the additional knowledge that red paint is some special kind of paint and that paint which contains metal pigments is harder to repair.*[37] *This TBox corresponds to the following normalized TBox:*[38]

$$\begin{aligned} RedMetallicCar &\equiv Car \sqcap \exists hasPainting.(RedPaint \sqcap \exists hasAdditive.MetalPigments) \\ RedPaint &\equiv \overline{RedPaint} \sqcap Paint \\ HardRepairablePaint &\equiv Paint \sqcap \exists hasAdditive.MetalPigments \end{aligned}$$

*The set of concept subsumption relationships that can be inferred from this TBox includes the fact that a red metallic car has a paint that is hard to repair, i.e.:*

$$RedMetallicCar \sqsubseteq \exists hasPainting.HardRepairablePaint \tag{55}$$

*Assume now that one wants to remove this consequence from the TBox. For this, one could first compute the set of laconic justifications for this entailment to identify the concepts' parts which are responsible for it.*[39] *This set includes the following laconic justification:*[40]

$$\begin{aligned} L_1 = \{&RedMetallicCar \sqsubseteq \exists hasPainting.RedPaint, \\ &RedPaint \sqsubseteq Paint, \\ &RedMetallicCar \sqsubseteq \exists hasPainting.\exists hasAdditive.MetalPigments, \\ &Paint \sqcap \exists hasAdditive.MetalPigments \sqsubseteq HardRepairablePaint\} \end{aligned}$$

*One can now select one implication from this laconic justification and remove it from the TBox to retract the consequence displayed in* (55)*.*[41] *The only reasonable implications to*

[37] Such a paint can display stripes if it is applied incorrectly, which is also known as the *venetian-blind effect*. This happens when the metal flakes take on a preferred orientation e.g. under the influence of gravity if the paint is sprayed on a vertically placed part [135].

[38] See Section 5.2 for information about the transformation of a generalized TBox into a normalized TBox.

[39] In the given case, the reasons for the entailment are admittedly obvious. However, one could think of examples where the relationships between the concepts are more complex and much harder to see.

[40] Remember that laconic justifications are defined based on the deductive closure of an ontology, as described in Section 4.2.1.

[41] Note that in this example all elements of the laconic justifications for the entailment (55) are related to the elements of the laconic justification $L_1$, i.e. there exists no fundamentally different way to infer (55).

*remove are those following from $RedMetallicCar$ and one could subsequently decide to use $\ominus$ to remove the consequence:*

$$RedMetallicCar \sqsubseteq \exists hasPainting.\exists hasAdditive.MetalPigments$$

*This gives the following equation where the result is derived in a similar way as in the previous example:*

$$\begin{aligned} & \mathcal{T}^*(RedMetallicCar) \ominus \exists hasPainting.\exists hasAdditive.MetalPigments \\ \equiv\ & Car \sqcap \exists hasPainting.(\overline{RedPaint} \sqcap Paint) \\ \equiv\ & Car \sqcap \exists hasPainting.RedPaint \end{aligned}$$

*With this result, the consequence displayed in* (55) *has been successfully removed.*

The following example shows how the order $\prec$ between concepts can be used to define preferred concepts for the retraction.

**Example 15** (Non-trivial conjunctions as subtrahends)**.** *Imagine an ontology that is used to orchestrate the daily activities in a hospital. This ontology contains nursing care plans for different types of patients and includes specific diet plans and medication plans.*[42] *Patients with diabetes for example have to follow a strict diet and are only served food with a low glycemic index. On the other hand, patients with a depressive disorder are allowed a full meal including white bread and cornflakes, but are given a morning dose of Nialamide, a nowadays obsolete antidepressant from the hydrazine class.*[43] *This is captured by the following axioms:*

$$\begin{aligned} DiabeticBreakfastMenu \equiv\ & \exists hasDish.BlueCheese \sqcap \exists hasDish.Porridge \\ & \sqcap \exists hasDish.WholeGrainBread \sqcap \exists hasDish.Eggs \\ FullBreakfastMenu \equiv\ & \exists hasDish.BlueCheese \sqcap \exists hasDish.Cornflakes \\ & \sqcap \exists hasDish.WhiteBread \sqcap \exists hasDish.Eggs \\ NursingPlanDepressionMorning \equiv\ & \exists ServeMenu.FullBreakfastMenu \\ & \sqcap \exists GiveMedicine.Nialamide \\ NursingPlanDiabetesMorning \equiv\ & \exists ServeMenu.DiabeticBreakfastMenu \\ & \sqcap \exists PerformProcedure.CheckBloodSugar \end{aligned}$$

*The morning distribution of medications is of higher importance than the availability of certain dishes. The ontology engineer reflected this in the arrangement of the order $\prec$ between concepts and specified that:*

$$ServeMenu \prec_{R_\mathcal{T}} GiveMedicine$$

[42] For the sake of simplicity it is assumed that each patient only suffers from one disease at a time.
[43] See https://pubchem.ncbi.nlm.nih.gov/compound/Nialamide.

*From the definition of $\prec$ (Definition 17) it follows that this order between role names is extended over all concepts such that for all concepts $C$ and $P$:*

$$\exists ServeMenu.C \prec \exists GiveMedicine.P \tag{56}$$

*In the 1960s, it turned out that the combination of Nialamide and blue cheese is highly dangerous: Nialamide, working as a monoamine oxidase inhibitor, hinders the metabolism of tyramine, an amine that can be found in blue cheese [136]. Assume that this has only recently become known. Reacting to this, the hospital now wants to remove the combination of these two from all nursing plans and performs the following operations:*

$$\mathcal{T}^*(NursingPlanDiabetesMorning) \ominus (\exists ServeMenu.\exists hasDish.BlueCheese \sqcap \exists GiveMedicine.Nialamide) \tag{57}$$

$$\mathcal{T}^*(NursingPlanDepressionMorning) \ominus (\exists ServeMenu.\exists hasDish.BlueCheese \sqcap \exists GiveMedicine.Nialamide) \tag{58}$$

*In the case of* (57)*, Nialamide is not a component of the nursing plan. It therefore follows from the definition of the operator (Definition 24) that the plan is not changed:*

$$\begin{aligned}&\mathcal{T}^*(NursingPlanDiabetesMorning) \ominus (\exists ServeMenu.\exists hasDish.BlueCheese \\ &\qquad \sqcap \exists GiveMedicine.Nialamide) \\ &= \mathcal{T}^*(NursingPlanDiabetesMorning)\end{aligned}$$

*However, in the case of* (58) *the combination can be found in the nursing plan:*

$$\begin{aligned}\mathcal{T}^*(NursingPlanDepressionMorning) \sqsubseteq\ &\exists ServeMenu.\exists hasDish.BlueCheese \\ &\sqcap \exists GiveMedicine.Nialamide\end{aligned}$$

*Since $\mathcal{T}^*(NursingPlanDepressionMorning)$ is a non-trivial conjunction of concepts, case 2 of the definition of the operator applies. The result of this case is given by:*

$$C \ominus P = \begin{cases} \prod_{i=1}^{n}(C_i \ominus P_{\prec}) & \text{if } C \sqsubseteq P \\ C & \text{otherwise} \end{cases}$$

*Note that the operation is pushed inside the conjuncts of the minuend. Note further that only the lowest conjunct of the subtrahend is retracted. From the previously defined order* (56) *is follows that:*

$$\exists ServeMenu.\exists hasDish.BlueCheese \prec \exists GiveMedicine.Nialamide$$

*The change of the nursing plan in* (58) *therefore results in the following equations, where for*

*reasons of readability the concepts are only expanded as much as necessary.*

$$\begin{aligned}
&\mathcal{T}^*(NursingPlanDepressionMorning) \ominus \exists ServeMenu.\exists hasDish.BlueCheese \\
&\qquad\qquad \sqcap \exists GiveMedicine.Nialamide \\
&= (\exists ServeMenu.\mathcal{T}^*(FullBreakfastMenu) \sqcap \exists GiveMedicine.Nialamide) \\
&\quad \ominus (\exists ServeMenu.\exists hasDish.BlueCheese \sqcap \exists GiveMedicine.Nialamide) \\
&= (\exists ServeMenu.\mathcal{T}^*(FullBreakfastMenu) \ominus \exists ServeMenu.\exists hasDish.BlueCheese) \\
&\quad \sqcap (\exists GiveMedicine.Nialamide \ominus \exists ServeMenu.\exists hasDish.BlueCheese)
\end{aligned}$$

*The remaining steps of the operation are similar to the previous example and are skipped here. The result is given by:*

$$\begin{aligned}
&(\exists ServeMenu.\mathcal{T}^*(FullBreakfastMenu) \ominus \exists ServeMenu.\exists hasDish.BlueCheese) \\
&\quad \sqcap (\exists GiveMedicine.Nialamide \ominus \exists ServeMenu.\exists hasDish.BlueCheese) \\
&\equiv \exists ServeMenu.(\exists hasDish.Cornflakes \sqcap \exists hasDish.WhiteBread \sqcap \exists hasDish.Eggs) \\
&\quad \sqcap \exists GiveMedicine.Nialamide
\end{aligned}$$

*With this the combination of Nialamide and blue cheese has been successfully removed from all nursing plans. In addition, the medication of all patients has been maintained while blue cheese is still served to patients that don't receive Nialamide.*

In the following section, it is shown that the operator has all the properties that were defined in Section 6.2.

### 7.4. Proofs

In this section it is proven that the operator fulfills all properties that were defined in Section 6.2. For this, some basic statements about the subsumption relationship between $\mathcal{EL}$ concepts are proven first. These are then used to show that the operator has the required properties.

**Lemma 3** (Existence of a model)**.** *For every concept $C$ there exists an interpretation $I$ such that:*

$$\emptyset \neq C^I$$

*Proof.* It is not possible to construct conflicting concept descriptions, because $\mathcal{EL}$ lacks negation [128]. Every concept has therefore a model. □

**Lemma 4.** *For concepts $C$, $P$ and $P'$:*

$$\text{If } C \not\sqsubseteq P, \text{ then } C \not\sqsubseteq P \sqcap P' \tag{59}$$

*Proof.* Assume that:

$$C \not\sqsubseteq P$$

From this it is implied that there exists an interpretation $I$ such that:

$$C^I \not\subseteq P^I$$

From set theory it follows that:

$$C^I \not\subseteq P^I \cap P'^I$$

This proves (59). □

**Lemma 5.** *For a concept $C$ and an atomic concept $A$ with $A \neq \top$ and $R \in \mathcal{R}_\mathcal{T}$:*

$$\exists R.C \not\sqsubseteq A$$

*Proof by contradiction.* Assume that:

$$\exists R.C \sqsubseteq A$$

Then for all interpretations $M$ it follows that:

$$(\exists R.C)^M \subseteq A^M \tag{60}$$

From Lemma 3 (existence of a model) it follows that there exists an interpretation $I$ such that:

$$(\exists R.C)^I \neq \emptyset$$

This implies that:

$$C^I \neq \emptyset$$

Now choose $y \in C^I$ and let $x$ be a new individual that does not occur in $\Delta^I$. Then construct a new model $J$ as follows:

$$\begin{aligned}
&\Delta^J = \Delta^I \cup \{x\} \\
&R^J = R^I \cup \{(x, y)\} \quad \text{if } R \neq u \\
&B^J = B^I \qquad\qquad \forall B \in \{Atoms(C) \cup A\} \backslash \{\top\} \\
&Q^J = Q^I \qquad\qquad \forall Q \in Roles(C) \backslash \{u, R\}
\end{aligned}$$

Note that:

$$A^J = A^I \tag{61}$$

Since no relations have been removed in $J$ compared to $I$ and no atomic concepts have been changed, from the semantics of $\mathcal{EL}$ it follows that:

$$C^I \subseteq C^J$$

But this means that:

$$y \in C^J$$

Note that $J$ was designed in such a way that:

$$(x, y) \in R^J$$

This implies that:

$$x \in (\exists R.C)^J$$

From (61) and $x \notin \Delta^I$ it can be concluded that:

$$x \notin A^J$$

But this means that:

$$(\exists R.C)^J \not\subseteq A^J$$

This contradicts (60). Therefore, $A$ cannot subsume $\exists R.C$. □

**Lemma 6.** *For a concept $C$ and $R \in \mathcal{R}_\mathcal{T}$:*

$$\exists R.C \sqsubseteq \exists u.C \tag{62}$$

*Proof.* The proof follows immediately from the semantics of $\mathcal{EL}$ and the definition of the universal role. For every interpretation $I$:

$$R^I \subseteq u^I$$

From this it follows that:

$$(\exists R.C)^I \subseteq (\exists u.C)^I$$

This implies (62). □

**Lemma 7.** *For the roles $C$ and $P$ with $R \in \mathcal{R}_\mathcal{T}$:*

$$\textit{If } C \sqsubseteq P\textit{, then } \exists R.C \sqsubseteq \exists R.P$$

*Proof.* This follows immediately from the semantics of $\mathcal{EL}$. □

**Lemma 8.** *For concepts $C$ and $P$ and a role $R$ with $u \notin Roles(P) \cup \{R\}$:*

$$\exists R.C \sqsubseteq \exists R.P \textit{ iff } C \sqsubseteq P \tag{63}$$

*Proof. For the direction "⇒": Proof by contradiction.* Assume that

$$\exists R.C \sqsubseteq \exists R.P \tag{64}$$

and

$$C \not\sqsubseteq P$$

Then there exists an interpretation $I$ such that:

$$C^I \not\subseteq P^I$$

Note that this implies that:

$$\emptyset \neq C^I \backslash P^I$$

Construct $J$ from $I$ as follows, with $x \notin \Delta^I$ and $y \in C^I \backslash P^I$:

$$\begin{aligned}
&\Delta^J = \Delta^I \cup \{x\} \\
&R^J = R^I \cup \{(x, y)\} \\
&A^J = A^I \qquad\qquad \forall A \in (Atoms(C) \cup Atoms(P)) \backslash \{\top\} \\
&Q^J = Q^I \qquad\qquad \forall Q \in (Roles(C) \cup Roles(P)) \backslash \{R, u\}
\end{aligned}$$

Note that:

$$y \notin P^I$$

Then from $y \in C^I$ and the construction of $J$ it follows that:

$$x \in (\exists R.C)^J \tag{65}$$

In addition, it follows from $y \notin P^I$ and the construction of $J$ that:[44]

$$y \notin P^J$$

From this it is implied that:

$$x \notin (\exists R.P)^J$$

This together with (65) leads to the conclusion that:

$$(\exists R.C)^J \not\subseteq (\exists R.P)^J$$

This contradicts (64). It therefore follows that:

$$\textit{If } \exists R.C \sqsubseteq \exists R.P \textit{, then } C \sqsubseteq P$$

*For the direction "⇐":* This direction follows from Lemma 7. This proves (63). □

**Lemma 9.** *For concepts $C$ and $P$ and $R \in \mathcal{R}_\mathcal{T} \backslash \{u\}$:*

$$\exists u.C \not\sqsubseteq \exists R.P \tag{66}$$

*Proof.* Assume that:

$$\exists u.C \sqsubseteq \exists R.P \tag{67}$$

From Lemma 3 (existence of a model) it follows that there exists an interpretation $I$ such that:

$$C^I \neq \emptyset \tag{68}$$

[44] Note that the universal role is not allowed to occur in $P$.

Let $x$ be a new individual that does not occur in $\Delta^I$. Then construct a new model $J$ as follows:

$$\begin{aligned}
\Delta^J &= \Delta^I \cup \{x\} \\
B^J &= B^I \qquad \forall B \in (Atoms(C) \cup Atoms(P)) \backslash \{\top\} \\
Q^J &= Q^I \qquad \forall Q \in (Roles(C) \cup Roles(\exists R.P)) \backslash \{u\}
\end{aligned}$$

Then from the semantics of $\mathcal{EL}$ and $R \neq u$ it follows that:

$$x \notin (\exists R.P)^J$$

From (68) and the construction of $J$ it follows that:

$$C^J \neq \emptyset$$

From this and the semantics of the universal role it follows that:

$$x \in (\exists u.C)^J$$

This contradicts (67) and proves (66). □

**Lemma 10.** *For the concepts $C, P, P'_1, P'_2, \ldots P'_n$ and $R \in \mathcal{R}_\mathcal{T}$ with $u \notin Roles(P)$:*

$$\exists R.C \sqsubseteq P \text{ iff } P = \top \text{ or } P = \prod_{i=1}^{n} \exists R.P'_i \text{ and } \forall i \in \{1, 2, \ldots, n\} : C \sqsubseteq P'_i$$

*Proof. For the direction "⇒": Proof by contradiction.* Assume that

$$\exists R.C \sqsubseteq P \tag{69}$$

In addition, assume that

$$P \neq \top$$

and $P$ has not the form described above. Then $P$ can have the following forms:

**Case 1:** $P$ is an atomic concept with $P \neq \top$,

**Case 2:** $P = \exists S.P'$ with a role $S \neq R$,

**Case 3:** $P = \exists R.P'$ with a concept $P'$ and $C \not\sqsubseteq P'$,

**Case 4:** $P = \prod_{i=1}^{n} P_i$ with $n > 1$, where every $P_i$ is a trivial conjunction and there exists $j$ such that one of the cases 1–3 can be applied to $P_j$.

The choice of these cases requires an explanation. First note that if $P$ is only a trivial conjunction, then the cases 1 – 3 cover all cases of $P$ that contradict the right side of the implication. The only case that then remains is the case that $P$ is a non-trivial conjunction. From the assumption that all concepts are normalized (Section 5.2) it is ensured that in this case $\top$ can not occur as a conjunct of $P$. This means that $P$ can only contain conjuncts that are atomic concepts different from top and conjuncts that are role restrictions. Now assume that $P$ only consist of conjuncts $P_i$ such that for all $i$ none of the cases 1 – 3 applies to $P_i$. Then choose one of these conjuncts and denote it by $X$. $X$ must be a role restriction, or else case 1 could be applied to $X$.[45]

[45] Remember that $\top$ cannot occur as a conjunct in $P$.

Second, the role of $X$ must equal $R$ or else case 2 could be applied to $X$. Last, if the role of $X$ is equal to $R$, then the filler of $X$ must subsume $C$ or else case 3 could be applied to $X$. But this means that $X = \exists R.P'$ with $C \sqsubseteq P'$. By an inductive argument, this holds true for all conjuncts $P_i$ and therefore

$$P = \prod_{i=1}^{n} \exists R.P_i' \text{ and } \forall i \in \{1, 2, \ldots, n\} : C \sqsubseteq P_i'$$

with concepts $P_i'$. This case however is captured by the right side of the lemma and cannot be used to disprove the implication. It can therefore be concluded that one of the cases 1 – 3 must be applicable to at least one conjunct of $P$ in order to disprove the lemma. This is captured by case 4.
In the following, it is now shown that the cases 1 – 4, together with (69), lead to a contradiction:

**Case 1:** $P$ is an atomic concept with $P \neq \top$. From Lemma 5 it follows that

$$\exists R.C \not\sqsubseteq P$$

This contradicts (69).

**Case 2:** $P = \exists S.P'$ with

$$S \neq R \tag{70}$$

From $u \notin Roles(P)$ it follows that:

$$S \neq u \tag{71}$$

From Lemma 3 (existence of a model) it follows that there exists an interpretation $I$ such that:

$$\emptyset \neq C^I$$

Choose

$$y \in C^I \tag{72}$$

and

$$x \notin \Delta^I \tag{73}$$

i.e. $x$ is a new individual that does not occur in the domain $\Delta^I$. Then construct a new model $J$ as follows:

$$\begin{aligned}
&\Delta^J = \Delta^I \cup \{x\} \\
&R^J = R^I \cup \{(x, y)\} \quad \textit{if } R \neq u \\
&A^J = A^I \qquad\qquad \forall A \in (Atoms(C) \cup Atoms(P)) \backslash \{\top\} \\
&Q^J = Q^I \qquad\qquad \forall Q \in (Roles(C) \cup Roles(P)) \backslash \{R, u\}
\end{aligned}$$

From the construction of $J$ it follows that:

$$S^J = S^I \tag{74}$$

From the construction of $J$ and (72) it is implied that:

$$x \in (\exists R.C)^J \tag{75}$$

From $S \neq R$ (70), $S \neq u$ (71), $x \notin \Delta^I$ (73) and $S^J = S^I$ (74) it follows that:

$$x \notin (\exists S.P')^J$$

From this and (75) it and can be concluded that:

$$(\exists R.C)^J \not\subseteq (\exists S.P')^J$$

With $P = \exists S.P'$ this implies that:

$$\exists R.C \not\sqsubseteq P$$

This contradicts (69).

**Case 3:** $P = \exists R.P'$ and $C \not\sqsubseteq P'$. From this, $u \notin Roles(P)$ and Lemma 8 it follows that:

$$\exists R.C \not\sqsubseteq \exists R.P'$$

This contradicts (69).

**Case 4:** $P = \bigsqcap_{i=1}^{n} P_i$ with $n > 1$, where every $P_i$ is a trivial conjunction and there exists a $j$ such that one case of the cases 1–3 can be applied to $P_j$. Then:
In the previous part of the proof it was shown that for all these cases it follows that:

$$\exists R.C \not\sqsubseteq P_j$$

But that means there exists a conjunct in $P$ that does not subsume $\exists R.C$. From this and Lemma 4 it follows that:

$$\exists R.C \not\sqsubseteq P$$

This contradicts (69).

This proves the direction "$\Rightarrow$".
*For the direction "$\Leftarrow$":* Consider the possible cases of $P$:

**Case 1:** $P = \top$. Then clearly $\exists R.C \sqsubseteq \top$.

**Case 2:** $P = \bigsqcap_{i=1}^{n} \exists R.P_i'$ and

$$\forall i \in \{1, 2, \ldots, n\} \colon C \sqsubseteq P_i' \tag{76}$$

Then the proof is given by a reductio ad absurdum argument. Assume that:

$$\exists R.C \not\sqsubseteq \bigsqcap_{i=1}^{n} \exists R.P_i' \tag{77}$$

From $u \notin \mathit{Roles}(P)$ it follows that:

$$R \neq u$$

From (77) it follows that there exists an interpretation $I$ such that:

$$(\exists R.C)^I \not\subseteq \bigcap_{i=1}^{n} (\exists R.P_i')^I \tag{78}$$

From (76) it is implied that:

$$\forall i \in \{1, 2, \ldots, n\} \colon C^I \subseteq P_i'^I$$

From this and the semantics of $\mathcal{EL}$ it follows that:

$$\forall i \in \{1, 2, \ldots, n\} \colon (\exists R.C)^I \subseteq (\exists R.P_i')^I$$

From this and set theory it follows that:

$$(\exists R.C)^I \subseteq \bigcap_{i=1}^{n} (\exists R.P_i')^I$$

This contradicts (78).

This proves the direction "$\Leftarrow$" and completes the proof of the lemma. □

**Lemma 11.** *For concepts $C$ and $P$:*

$$\textit{If } C \sqsubseteq P\textit{, then } C \sqsubseteq f_{\mathit{contract}}(P).$$

*Proof.* This follows immediately from the definition of $f_{\mathit{contract}}$ (Definition 23 in Section 7.1), because the result of $f_{\mathit{contract}}(P)$ is either $P$ or $\top$. □

**Theorem 1** (Closedness)**.** *For concepts $C$ and $P$:*

$$C \ominus P \textit{ is a concept}$$

*Proof.* This follows immediately from the well-definedness of the operator (Lemma 2 in Section 7.2) and its definition. □

**Theorem 2** (Preservation). *For concepts $C$, $P$ and $P'$:*

$$\text{If } P \equiv P', \text{ then } C \ominus P \equiv C \ominus P' \tag{79}$$

*Proof.* Assume that:

$$P \equiv P'$$

From this, the assumption that all concepts are normalized (Section 5.2) and the existence of a unique normal form for equivalent concepts (Lemma 1 in Section 5.1) it can be concluded that:

$$P = P'$$

From this (79) follows. □

**Lemma 12.** *For concepts $C_1, C_2, \ldots, C_n$ and $P$, where $P$ is only a trivial conjunction with a single conjunct, with $u \notin Roles(P)$:*

$$\text{If } \forall i \in \{1, 2, \ldots, n\} \colon C_i \not\sqsubseteq P, \text{ then } C_1 \sqcap C_2 \sqcap \ldots \sqcap C_n \not\sqsubseteq P \tag{80}$$

*Proof.* Assume that:

$$\forall i \in \{1, 2, \ldots, n\} \colon C_i \not\sqsubseteq P$$

Then for each $C_i$ there exists an interpretation $J_i$ such that:

$$C_i^{J_i} \not\subseteq P^{J_i}$$

This implies that each $C_i^{J_i}$ contains an element that does not occur in $P^{J_i}$. Let these elements be denoted by $a_i$:

$$\forall i \in \{1, 2, \ldots, n\} \; \exists a_i \in \Delta^{J_i} \colon a_i \in C_i^{J_i} \wedge a_i \notin P^{J_i} \tag{81}$$

Let w.l.o.g.:

$$\bigcap_{i=1}^{n} \Delta^{J_i} = \emptyset$$

If the domains are not disjoint, rename the elements of the interpretations accordingly. Choose $b$ such that $b$ does not occur in the domain of any interpretation $J_i$, i.e.:

$$b \notin \bigcup_{i=1}^{n} \Delta^{J_i}$$

Let

$$Atoms = (\bigcup_{i=1}^{n} Atoms(C_i) \cup Atoms(P)) \backslash \{\top\}$$

denote all atomic concepts that occur in at least one of the concepts $C_i$ and $P$, excluding the top concept. Let

$$Roles = (\bigcup_{i=1}^{n} Roles(C_i) \cup Roles(P)) \backslash \{u\}$$

denote all roles that occur in at least one of the concepts $C_i$ and $P$, excluding the universal role. Then construct a new interpretation $L$ as follows:

$$\Delta^L = \bigcup_{i=1}^{n} \Delta^{J_i} \cup \{b\}$$

$$\forall A \in Atoms\colon A^L = \begin{cases} \bigcup_{i=1}^{n} A^{J_i} \cup \{b\} & \text{if } \exists i \in \{1, 2, \ldots, n\}\colon a_i \in A^{J_i} \\ \bigcup_{i=1}^{n} A^{J_i} & \text{otherwise} \end{cases}$$

$$\forall R \in Roles\colon R^L = \bigcup_{i=1}^{n} (R^{J_i} \cup \{(b, x) \mid x \in \Delta^{J_i} \wedge (a_i, x) \in R^{J_i}\})$$

First, note that:

$$\forall i \in \{1, 2, \ldots, n\}\colon b \in C_i^L$$

This follows immediately from the construction of $L$ and (81). Consequently:

$$b \in \bigcap_{i=1}^{n} C_i^L \tag{82}$$

On the other hand:

$$b \notin P^L \tag{83}$$

This follows again from the construction of $L$ and (81) and is explained in the following: Assume that $P$ is an atomic concept. Then $b$ could only be present in $P^L$ if an $a_i$ is present in $P^{J_i}$.[46] This is not possible because of the choice of $a_i$ in (81). The same holds true if $P$ is a role restriction.[47] By an inductive argument, $b$ is also not an element of $P^L$ if $P$ is a non-trivial conjunction. From (82) and (83) it then follows that:

$$\bigcap_{i=1}^{n} C_i^L \not\sqsubseteq P^L$$

This proves (80). □

**Lemma 13.** *For concepts $C_1, C_2, \ldots, C_m, P_1, \ldots, P_n$ that are all trivial conjunctions, with $u \notin \bigcup_{i=1}^{n} Roles(P_i)$:*

$$\begin{aligned} &\textit{If } \exists j \in \{1, 2, \ldots, n\}\, \forall i \in \{1, 2, \ldots, m\}\colon C_i \not\sqsubseteq P_j, \\ &\quad \textit{then } C_1 \sqcap C_2 \sqcap \ldots \sqcap C_m \not\sqsubseteq P_1 \sqcap P_2 \sqcap \ldots \sqcap P_n \end{aligned}$$

[46] Note again that the domains $\Delta^{J_i}$ are disjoint.
[47] Note that the universal role is not allowed to occur within $P$.

*Proof.* Assume that:

$$\exists j \in \{1, 2, \ldots, n\}\, \forall i \in \{1, 2, \ldots, m\} \colon C_i \not\sqsubseteq P_j$$

From Lemma 12 it follows that:

$$C_1 \sqcap C_2 \sqcap \ldots \sqcap C_m \not\sqsubseteq P_j$$

From this and Lemma 4 it can be concluded that:

$$C_1 \sqcap C_2 \sqcap \ldots \sqcap C_m \not\sqsubseteq P_1 \sqcap P_2 \sqcap \ldots \sqcap P_n$$

□

**Lemma 14.** *For concepts $C_1, C_2, \ldots, C_n, P_1, P_2, \ldots, P_m$ with $u \notin \bigcup_{i=1}^{m} Roles(P_i)$:*

$$\prod_{i=1}^{n} C_i \sqsubseteq \prod_{j=1}^{m} P_j \;\; \textit{iff} \;\; \forall j \in \{1, 2, \ldots, m\}\, \exists i \in \{1, 2, \ldots, n\} \colon C_i \sqsubseteq P_j$$

*Proof by contradiction. For the direction "⇒":* Assume that

$$\prod_{i=1}^{n} C_i \sqsubseteq \prod_{j=1}^{m} P_j \tag{84}$$

and

$$\exists j \in \{1, 2, \ldots, m\}\, \forall i \in \{1, 2, \ldots, n\} \colon C_i \not\sqsubseteq P_j$$

From this and Lemma 13 it follows that:

$$\prod_{i=1}^{n} C_i \not\sqsubseteq \prod_{j=1}^{m} P_j$$

This contradicts (84) and proves the direction "⇒".
*For the direction "⇐":* Assume that

$$\forall j \in \{1, 2, \ldots, m\}\, \exists i \in \{1, 2, \ldots, n\} \colon C_i \sqsubseteq P_j \tag{85}$$

and

$$\prod_{i=1}^{n} C_i \not\sqsubseteq \prod_{j=1}^{m} P_j$$

Then there exists an interpretation $I$ such that:

$$(\prod_{i=1}^{n} C_i)^I \not\subseteq (\prod_{j=1}^{m} P_j)^I \tag{86}$$

Let $S_C$ denote the set of all conjuncts $C_i$ which are subsumed by a conjunct $P_k$, i.e.:

$$S_C = \{C_i \mid i \in \{1, 2, \ldots, n\} \textit{ and } \exists k \in \{1, 2, \ldots, m\} \colon C_i \sqsubseteq P_k\} \tag{87}$$

From this and (85) it follows that for all interpretations $J$:

$$\forall j \in \{1, 2, \ldots, m\}\ \exists C' \in S_C \colon C'^J \subseteq P_j^J$$

From this and set theory it is implied that:

$$\bigcap_{C' \in S_C} C'^J \subseteq \bigcap_{j=1}^{m} P_j^J$$

From this and (87) it follows that:

$$\bigcap_{i=1}^{n} C_i^J \subseteq \bigcap_{C' \in S_C} C'^J \subseteq \bigcap_{j=1}^{m} P_j^J$$

This contradicts (86) and proves the direction "$\Leftarrow$". This proves the lemma. □

**Definition 26** (Minimal subsumed set)**.** *Let $C_1, C_2, \ldots, C_n, P_1, P_2, \ldots, P_m$ be concepts that are trivial conjunctions and let $\bigsqcap_{i=1}^{n} C_i \sqsubseteq \bigsqcap_{j=1}^{m} P_j$. Then $S$ is a minimal subsumed set of $\bigsqcap_{i=1}^{n} C_i$ w.r.t. $\bigsqcap_{j=1}^{m} P_j$ if $S$ contains only indices of conjuncts $C_i$ such that*

$$\bigsqcap_{i \in S} C_i \sqsubseteq \bigsqcap_{j=1}^{m} P_j \tag{88}$$

*and for every other set $S'$ such that*

$$S' \neq \emptyset$$

*and*

$$|S'| < |S|$$

*i.e. $S'$ is not empty and has less elements than $S$, it follows that:*

$$\bigsqcap_{i \in S'} C_i \not\sqsubseteq \bigsqcap_{i=1}^{m} P_i$$

*In other words, if the removal of one element from $S$ turns $S$ empty or destroys the subsumption relationship in* (88)*, then $S$ is a minimal subsumed set of $\bigsqcap_{i=1}^{n} C_i$ w.r.t. $\bigsqcap_{j=1}^{m} P_j$.*

**Lemma 15.** *Let $\bigsqcap_{i=1}^{n} C_i \sqsubseteq \bigsqcap_{j=1}^{m} P_j$, where $C_i$ and $P_j$ are trivial conjunctions of concepts and let $S$ be a minimal subsumed set of $\bigsqcap_{i=1}^{n} C_i$ w.r.t. $\bigsqcap_{j=1}^{m} P_j$. Then:*

$$\forall j \in \{1, 2, \ldots, m\}\ \exists i \in S \colon C_i \sqsubseteq P_j \tag{89}$$

*and*

$$\forall i \in S\ \exists j \in \{1, 2, \ldots, m\} \colon C_i \sqsubseteq P_j \tag{90}$$

In other words, every conjunct of $C$ that is in the set $S$ is subsumed by a conjunct of $P$ and there does not exist a conjunct of $P$ which does not subsume a conjunct that is in the set $S$.

*Proof.* From Definition 26 it follows that:

$$\bigsqcap_{i \in S} C_i \sqsubseteq \bigsqcap_{j=1}^{m} P_j \tag{91}$$

From this and Lemma 14 it follows that every conjunct in $\bigsqcap_{j=1}^{m} P_j$ subsumes at least one conjunct $C_i$, with $i \in S$. This proves (89). The following part of the proof is given by a reductio ad absurdum argument: Assume that there exists a conjunct $C_k$, with $k \in S$, that is not subsumed by a conjunct in $\bigsqcap_{j=1}^{m} P_j$. From this, Lemma 14 and (91) it follows that:

$$|S| > 1 \tag{92}$$

Construct a new set $S'$ from $S$ such that:

$$S' = S \backslash \{k\}$$

From this and (92) it follows that

$$S' \neq \emptyset \tag{93}$$

and

$$|S'| < |S| \tag{94}$$

From the construction of $S'$, (89) and Lemma 14 it is implied that:

$$\bigsqcap_{j \in S'} C_j \sqsubseteq \bigsqcap_{j=1}^{m} P_j$$

But from this, (93) and (94) it follows that according to Definition 26 $S$ cannot be a minimal subsumed set, because it is not minimal. This is a contradiction and therefore proves (90). □

**Theorem 3** (Inclusion). *For concepts $C$ and $P$:*

$$C \sqsubseteq C \ominus P \tag{95}$$

*Proof by induction.* The proof is given by an induction over the structure of $C$.

**Induction base:** $C$ is an atomic concept. Then from the definition of the operator (Definition 24) the inclusion property (95) follows immediately.

**Induction hypothesis:** The inclusion property $C' \sqsubseteq C' \ominus X$ holds for all parts $C'$ of $C$, where $X$ is a concept such that $u \notin Roles(X)$.[48]

**Induction step:** It is now shown that the inclusion property holds for $C$. Consider the following cases for $C$:

**Case 1:** $C$ is a non-trivial conjunction of concepts $C = C_1 \sqcap C_2 \sqcap \ldots \sqcap C_n$ such that each $C_i$ with $i \in \{1, 2, \ldots, n\}$ is a trivial conjunction. Then the induction hypothesis implies that for every concept $P'$ such that $u \notin Roles(P')$:

$$\forall i \in \{1, 2, \ldots, n\} \colon C_i \sqsubseteq C_i \ominus P' \tag{96}$$

Consider the following case distinction between the results of $C \ominus P$:

**Case 1.1:** $C \ominus P = C$. Then from this (95) follows.

**Case 1.2:** $C \ominus P = \bigsqcap_{i=1}^{n} (C_i \ominus P_{\prec})$ where $P_{\prec}$ is the lowest conjunct in $P$. Then from (96) it follows that every conjunct in $C \ominus P$ subsumes a conjunct $C_i$ of $C$. But this means that Lemma 14 can be applied and it can be concluded that:

$$\bigsqcap_{i=1}^{n} C_i \sqsubseteq \bigsqcap_{i=1}^{n} (C_i \ominus P_{\prec})$$

From this the inclusion property (95) follows.

**Case 2:** $C = \exists R.C_1$ with $R \in \mathcal{R}_{\mathcal{T}}$ and a concept $C_1$. Then the induction hypothesis implies that for every concept $P'$ such that $u \notin Roles(P')$:

$$C_1 \sqsubseteq C_1 \ominus P' \tag{97}$$

From the definition of $\ominus$ (Definition 24 in Section 7.1) it follows that the following cases are possible:

**Case 2.1:** $C \ominus P = \top$ or $C \ominus P = C$. Then it immediately follows that:

$$C \sqsubseteq C \ominus P$$

[48] Note that from the assumptions in Section 5.2 it follows that the universal role is not allowed to occur in $P$.

**Case 2.2:** $C \ominus P = f_{contract}(\exists R.(C_1 \ominus P_1))$ where $P_{\prec} = \exists R.P_1$. Then:
From (97) and Lemma 8 it follows that:

$$\exists R.C_1 \sqsubseteq \exists R.(C_1 \ominus P_1)$$

From this and Lemma 11 it follows that:

$$\exists R.C_1 \sqsubseteq f_{contract}(\exists R.(C_1 \ominus P_1))$$

This proves (95). □

**Lemma 16.** *For a concept $P$ with $u \notin Roles(P)$:*

$$\top \ominus P = \top$$

*Proof.* This follows immediately from the definition of $\ominus$ (Definition 24 in Section 7.1). □

**Theorem 4** (Destruction)**.** *For a concept $C$:*

$$C \ominus \top \equiv \top \tag{98}$$

*Proof.* The proof is given by an induction over the structure of $C$.

**Induction base:** $C$ is an atomic concept. Then:
Since $C \sqsubseteq \top$ is true for every concept $C$, it follows from the definition of $\ominus$ (Definition 24 in Section 7.1) that:

$$C \ominus \top = \top$$

This implies (98).

**Induction hypothesis:** The destruction property $C' \ominus \top \equiv \top$ holds for all parts $C'$ of $C$.

**Induction step:** It is now shown that the destruction property holds for $C$. Consider the following cases for $C$:

**Case 1:** $C$ is a non-trivial conjunction of concepts $C_i$ of the form $C = C_1 \sqcap C_2 \sqcap \ldots \sqcap C_n$, such that each $C_i$ with $i \in \{1, 2, \ldots, n\}$ is a trivial conjunction. The induction hypothesis implies that:

$$\forall i \in \{1, 2, \ldots, n\} \colon C_i \ominus \top \equiv \top \tag{99}$$

From the definition of $\ominus$ (Definition 24 in Section 7.1), $C \sqsubseteq \top$ and the fact that $\top$ is the lowest conjunct in $\top$ it follows that:

$$C \ominus \top = \prod_{i=1}^{n} (C_i \ominus \top)$$

From (99) it follows that:

$$\bigsqcap_{i=1}^{n}(C_i \ominus \top) = \bigsqcap_{i=1}^{n} \top$$

From this it can be concluded that:

$$\bigsqcap_{i=1}^{n} \top \equiv \top$$

This implies (98).

**Case 2:** $C = \exists R.C_1$, with a concept $C_1$ and $R \in \mathcal{R}_\mathcal{T}$. Then from $\exists R.C_1 \sqsubseteq \top$ and the definition of $\ominus$ (Definition 24 in Section 7.1) it follows that:

$$\exists R.C_1 \ominus \top = \top$$

This implies (98).

This proofs the destruction property (98). □

**Theorem 5** (Vacuity)**.** *For concepts $C$ and $P$:*

$$\textit{If } C \not\sqsubseteq P\textit{, then } C \ominus P = C$$

*Proof.* The proof follows immediately from the definition of $\ominus$ (Definition 24 in Section 7.1). □

**Lemma 17.** *For a concept $C$ and the lowest conjunct $C_\prec$ of $C$:*

$$\textit{If } C \not\equiv \top\textit{, then } C_\prec \not\equiv \top$$

*Proof.* The proof follows from the semantics of $\mathcal{EL}$ and the definition of $\prec$ (Definition 17). □

**Theorem 6** (Success)**.** *For concepts $C$ and $P$:*

$$\textit{If } \top \not\equiv P\textit{, then } (C \ominus P) \not\sqsubseteq P \tag{100}$$

*Proof.* Assume that:

$$\top \not\equiv P \tag{101}$$

Then consider the following cases:

**Case 1:**

$$C \not\sqsubseteq P \tag{102}$$

From the vacuity property (Theorem 5) it follows that

$$C \ominus P = C$$

From this and (102) the success property (100) follows.

**Case 2:**

$$C \sqsubseteq P \tag{103}$$

The proof is given by an induction over the structure of $C$.

**Induction base:** $C$ is an atomic concept. Then from (103) and the definition of $\ominus$ (Definition 24) it follows that:

$$C \ominus P \equiv \top$$

Together with $\top \not\equiv P$ (101) this implies (100).

**Induction hypothesis:** The success property

$$\textit{If } \top \not\equiv X\textit{, then } (C' \ominus X) \not\sqsubseteq X$$

holds for all parts $C'$ of $C$, where $X$ is a concept such that $u \notin Roles(X)$.[49]

**Induction step:** It is now shown that the success property holds for $C$. Consider the following cases for $C$:

**Case 1:** $C$ is a non-trivial conjunction of concepts $C = C_1 \sqcap C_2 \sqcap \ldots \sqcap C_n$, such that each $C_i$ with $i \in \{1, 2, \ldots, n\}$ is a trivial conjunction. Then the induction hypothesis implies that for every conjunct $C_i$ with $i \in \{1, 2, \ldots, n\}$ and every concept $P'$ with $u \notin Roles(P')$:

$$\textit{If } \top \not\equiv P'\textit{, then } (C_i \ominus P') \not\sqsubseteq P' \tag{104}$$

From the definition of $\ominus$ (Definition 24 in Section 7.1) and the assumption $C \sqsubseteq P$ (103) it follows that

$$C \ominus P \equiv \prod_{i=1}^{n} (C_i \ominus P_{\prec}) \tag{105}$$

where $P_{\prec}$ denotes the lowest conjunct in $P$. From $\top \not\equiv P$ (101) and Lemma 17 it follows that

$$P_{\prec} \not\equiv \top$$

The application of (104) to (105) results in:

$$\forall i \in \{1, 2, \ldots, n\} \colon C_i \ominus P_{\prec} \not\sqsubseteq P_{\prec}$$

But this means that there exists a conjunct in $P$ that does not subsume any conjunct of $C \ominus P$. From this and Lemma 14 the success property (100) follows.

[49] Note that from the assumptions in Section 5.2 it follows that the universal role is not allowed to occur in $P$.

**Case 2:** $C = \exists R.C_1$ with a concept $C_1$ and $R \in \mathcal{R}_\mathcal{T}$. Then the induction hypothesis implies that for every concept $P'$ with $u \notin Roles(P')$:

$$\textit{If } \top \not\equiv P', \textit{ then } C_1 \ominus P' \not\sqsubseteq P' \tag{106}$$

From $u \notin Roles(P)$, $\top \not\equiv P$ (101), $C \sqsubseteq P$ (103) and Lemma 10 it follows that:

$$P = \prod_{i=1}^{m} \exists R.P_i \tag{107}$$

Assume w.l.o.g. that:

$$P_\prec = \exists R.P_1$$

From this, $C \sqsubseteq P$ (103) and the definition of $\ominus$ it follows that:

$$C \ominus P = f_{contract}(\exists R.(C_1 \ominus P_1))$$

Now consider the following cases for the result of $C_1 \ominus P_1$:

**Case 2.2.1:**

$$C_1 \ominus P_1 \equiv \top$$

Then:
From this and the definition of $f_{contract}$ (Definition 23 in Section 7.1) it follows that:

$$f_{contract}(\exists R.(C_1 \ominus P_1)) = \top$$

From this and (101) the success property (100) follows.

**Case 2.2.2:**

$$C_1 \ominus P_1 \not\equiv \top \tag{108}$$

Then:
From the destruction property (Theorem 4) it follows that:

$$P_1 \not\equiv \top$$

From the normalization of all concepts and $u \notin Roles(P_1)$ it is implied that:

$$P_1 \not\equiv \top \tag{109}$$

From (108) and the definition of $f_{contract}$ (Definition 23 in Section 7.1) it follows that:

$$f_{contract}(\exists R.(C_1 \ominus P_1)) = \exists R.(C_1 \ominus P_1)$$

From (109) and (106) it follows that:

$$C_1 \ominus P_1 \not\sqsubseteq P_1$$

From this, (107), $u \notin Roles(P)$ and Lemma 10 it follows that:

$$\exists R.(C_1 \ominus P_1) \not\sqsubseteq \exists R.P_1$$

From this and Lemma 4 it follows that:

$$\exists R.(C_1 \ominus P_1) \not\sqsubseteq \prod_{i=1}^{m} \exists R.P_i$$

This together with (107) proves the success property (100).

□

**Lemma 18.** *Let $P = \prod_{i=1}^{n} \exists R.P_i$, and $P' = \prod_{j=1}^{m} \exists R.P'_j$ be concepts such that $P$ and $P'$ are independent, where $R$ is a role including the universal role. Let $k$ denote the index of the lowest conjunct in $\prod_{i=1}^{n} \exists R.P_i$. Then:*

$$\forall i \in \{1, 2, \ldots, m\} \colon P_k \text{ and } P'_i \text{ are independent} \tag{110}$$

*Proof by contradiction.* Assume there exists an index $l \in \{1, 2, \ldots, m\}$ such that $P_k$ and $P'_l$ are not independent.[50] Let w.l.o.g.:

$$P_k \sqsubseteq P'_l$$

It follows that:

$$\exists R.P_k \sqsubseteq \exists R.P'_l$$

This contradicts the assumption that $P$ and $P'$ are independent and proves (110). □

**Lemma 19.** *For concepts $C$, $P$ and $P'$:*

$$\text{If } C \sqsubseteq P, \text{ then } C \sqcap P' \sqsubseteq P$$

*Proof.* The proof follows immediately from the semantics of $\mathcal{EL}$. □

**Lemma 20.** *For the atomic concept $A$ and the concepts $P$ and $P'$, such that $u \notin Roles(P) \cup Roles(P')$:*

$$\text{If } A \sqsubseteq P \text{ and } A \sqsubseteq P', \text{ then } P \text{ and } P' \text{ are not independent}$$

*Proof.* Assume that:

$$A \sqsubseteq P \tag{111}$$

Assume also that:

$$A \sqsubseteq P' \tag{112}$$

From (111) it follows that $P$ is either equivalent to $A$ or $P \equiv \top$.[51] In the same way, if $A \sqsubseteq P'$ (112), $P'$ is either equivalent to $A$ or $P' \equiv \top$. In all these cases, $P$ and $P'$ are not independent. □

[50] See again Definition 20 in Section 6.2.2 for the definition of independent concepts.
[51] Note that all concepts are normalized.

**Theorem 7** (Independence). *For concepts $C$, $P$ and $P'$:*

$$\textit{If } C \sqsubseteq P' \textit{ and } P \textit{ and } P' \textit{ are independent, then } C \ominus P \sqsubseteq P' \tag{113}$$

*Proof.* Assume that:

$$C \sqsubseteq P' \tag{114}$$

Assume also that:

$$P \text{ and } P' \text{ are independent} \tag{115}$$

In addition, let $P$ w.l.o.g. be a possibly trivial conjunction of $n$ conjuncts that are trivial conjunctions. Let $P'$ w.l.o.g. be a possibly trivial conjunction of $m$ conjuncts that are trivial conjunctions. For $C \not\sqsubseteq P$ it follows from the vacuity property (Theorem 5) that:

$$C \ominus P = C$$

From this and (114) the independence property in (113) follows. Assume now that:

$$C \sqsubseteq P \tag{116}$$

The proof is then given by an induction over the structure of $C$.

**Induction base:** $C$ is an atomic concept. Then from this, (114), (115) and Lemma 20 it follows that:

$$C \not\sqsubseteq P$$

From this and the vacuity property (Theorem 5) it follows that:

$$C \ominus P = C$$

From this and (114) it can be concluded that:

$$C \ominus P \sqsubseteq P'$$

**Induction hypothesis:** The independence property holds for all parts $C'$ of $C$, i.e.:

$$\textit{If } C' \sqsubseteq X \textit{ and } X \textit{ and } X' \textit{ are independent, then } C' \ominus X' \sqsubseteq X$$

where $X$ and $X'$ are concepts such that $u \notin \mathit{Roles}(X')$.[52]

**Induction step:** It is now shown that the independence property holds for $C$. Consider the following cases for $C$.

[52] Note that from the assumptions in Section 5.2 it follows that the universal role is not allowed to occur in $P$.

**Case 1:** $C$ is a non-trivial conjunction of concepts $\bigsqcap_{i=1}^{n} C_i$, where each $C_i$ is a trivial conjunction. Then the induction hypothesis implies that for independent concepts $X$ and $X'$ such that $u \notin \mathit{Roles}(X')$:

$$\forall i \in \{1, 2, \ldots, n\}\colon \textit{If } C_i \sqsubseteq X, \textit{ then } C_i \ominus X' \sqsubseteq X \tag{117}$$

Let $P'_i$ denote the $i$th conjunct of $P'$. From (116) and the definition of $\ominus$ it follows that:

$$C \ominus P = \bigsqcap_{i=1}^{n} (C_i \ominus P_{\prec}) \tag{118}$$

From $C \sqsubseteq P'$ (114) and Definition 26 it follows that there exists a set $S$ of indices with

$$(\bigsqcap_{j \in S} C_j) \sqsubseteq P'$$

that is minimal in size. But that means that Lemma 15 implies that:

$$\forall i \in \{1, 2, \ldots, m\}\ \exists j \in S\colon C_j \sqsubseteq P'_i \tag{119}$$

Note again that $m$ denote the number of trivial conjuncts in $P'$. From the assumption that $P$ and $P'$ are independent (115) and the definition of independence (Definition 20 in Section 6.2.2) it is implied that:

$$\forall i \in \{1, 2, \ldots, m\}\colon\ P_{\prec} \textit{ and } P'_i \textit{ are independent} \tag{120}$$

But this means that (117) can be applied to (120) and it can be concluded that:

$$\forall i \in \{1, 2, \ldots, m\}\ \forall j \in S\colon \textit{If } C_j \sqsubseteq P'_i, \textit{ then } C_j \ominus P_{\prec} \sqsubseteq P'_i$$

From this and (119) it follows that every conjunct in $P'$ subsumes a conjunct in $\bigsqcap_{j \in S}(C_j \ominus P_{\prec})$. From this and Lemma 14 it is implied that:

$$\bigsqcap_{j \in S} (C_j \ominus P_{\prec}) \sqsubseteq P'$$

From this, the fact that $S$ contains only indices in the range from 1 to $n$ and Lemma 19 it follows that:

$$\underbrace{\bigsqcap_{j=1}^{n} (C_j \ominus P_{\prec})}_{C \ominus P} \sqsubseteq P'$$

This together with (118) implies that:[53]

$$C \ominus P \sqsubseteq P'$$

[53] Note that $n$ denotes the number of conjuncts in $C$.

**Case 2:** $C$ is of the form $C = \exists R.C_1$ with a role $R$ and a concept $C_1$. Then the induction hypothesis implies that for independent concepts $X$ and $X'$ such that $u \notin \mathit{Roles}(X')$:

$$\textit{If } C_1 \sqsubseteq X, \textit{ then } C_1 \ominus X' \sqsubseteq X \tag{121}$$

From the fact that $P$ and $P'$ are independent (115) it is implied that

$$P \not\equiv \top \tag{122}$$

and

$$P' \not\equiv \top \tag{123}$$

From this and the definition of the order $\prec$ (Definition 17) it follows that:

$$P_{\prec} \neq \top$$

From $P \not\equiv \top$ (122) and the normalization of all concepts it is implied that:

$$P \neq \top$$

From this, $C \sqsubseteq P$ (114), $C = \exists R.C_1$ and Lemma 10 it follows that

$$P = \prod_{i=1}^{n} \exists R.P_i \tag{124}$$

and that

$$\forall i \in \{1, 2, \ldots, n\} \colon C_1 \sqsubseteq P_i$$

From $P' \not\equiv \top$ (123) it is implied that:

$$P' \neq \top$$

From this, $C \sqsubseteq P'$ (114), $C = \exists R.C_1$ and Lemma 10 it follows that

$$P' = \prod_{j=1}^{m} \exists R.P'_j \tag{125}$$

and

$$\forall j \in \{1, 2, \ldots, m\} \colon C_1 \sqsubseteq P'_j \tag{126}$$

Let w.l.o.g.:

$$P_{\prec} = \exists R.P_1 \tag{127}$$

From this, $C \sqsubseteq P$ (116), $P \not\equiv \top$ (122) and the definition of the operator (Definition 24) it follows that:

$$C \ominus P = f_{contract}(\exists R.(C_1 \ominus P_1)) \tag{128}$$

From the fact that $P$ and $P'$ are independent (115), (124), (125), (127) and Lemma 18 it follows that:

$$\forall j \in \{1, 2, \ldots, m\} \colon P_1 \textit{ and } P'_j \textit{ are independent} \tag{129}$$

This implies that:

$$\forall j \in \{1, 2, \ldots, m\} \colon P'_j \not\equiv \top \tag{130}$$

From (121), (126) and (129) it can be concluded that:

$$\forall j \in \{1, 2, \ldots, m\} \colon C_1 \ominus P_1 \sqsubseteq P'_j \tag{131}$$

From this and (130) it follows that:

$$C_1 \ominus P_1 \not\equiv \top$$

From this and the definition of $\ominus$ (Definition 24) it can be concluded that:

$$f_{contract}(\exists R.(C_1 \ominus P_1)) = \exists R.(C_1 \ominus P_1) \tag{132}$$

From (131) and Lemma 7 it follows that:

$$\forall j \in \{1, 2, \ldots, m\} \colon \exists R.(C_1 \ominus P_1) \sqsubseteq \exists R.P'_j$$

From this and (132) it follows that:

$$\forall j \in \{1, 2, \ldots, m\} \colon \underbrace{f_{contract}(\exists R.(C_1 \ominus P_1))}_{C \ominus P} \sqsubseteq \exists R.P'_j$$

From this and (128) it follows that:

$$\forall j \in \{1, 2, \ldots, m\} \colon C \ominus P \sqsubseteq \exists R.P'_j$$

From this and $P' = \bigsqcap_{j=1}^{m} \exists R.P'_j$ (125) it follows that all conjuncts in $P'$ subsume $C \ominus P$. From this and set theory it can be concluded that:

$$C \ominus P \sqsubseteq P'$$

□

**Lemma 21.** *For concepts $C$, $P$ and $P'$:*

$$\textit{If } C \sqsubseteq P \sqcap P', \textit{ then } C \sqsubseteq P$$

*Proof.* This follows immediately from the semantics of $\mathcal{EL}$. □

**Theorem 8** (Informational economy)**.**
*For concepts $C$ and $P$:*

$$\text{If } C \sqsubseteq P, \text{ then } C \ominus P = C \ominus P_{\prec} \tag{133}$$

*Proof.* Assume that:

$$C \sqsubseteq P \tag{134}$$

If $P$ is a trivial conjunction then it follows that:

$$P = P_{\prec}$$

This implies (133). Assume in the following that $P$ is a non-trivial conjunction. If

$$P \equiv \top$$

then (133) follows immediately from the normalization of all concepts and the definition of the order $\prec$.[54] Assume in the following that:

$$P \not\equiv \top \tag{135}$$

In addition, assume w.l.o.g. that $P$ has the form $P = P_1 \sqcap P_2 \sqcap \ldots \sqcap P_n$ where each $P_i$ is a trivial conjunction. Then:
From (134) and Lemma 21 it follows that:

$$C \sqsubseteq P_{\prec} \tag{136}$$

Consider the following cases for $C$:

**Case 1:** $C$ is an atomic concept. Then:
From (134) and the definition of $\ominus$ (Definition 24) it follows that:

$$C \ominus P = \top \tag{137}$$

From (136) and the definition of $\ominus$ it follows that:

$$C \ominus P_{\prec} = \top$$

From this and (137) it can be concluded that:

$$C \ominus P = C \ominus P_{\prec}$$

This implies (133).

[54] See again Definition 16 and Definition 17 in Section 5.1.

**Case 2:** $C$ is a non-trivial conjunction $C = \bigsqcap_{i=1}^{m} C_i$ where each $C_i$ is a trivial conjunction. Then:
From $C \sqsubseteq P$ (134) and the definition of $\ominus$ (Definition 24) it follows that:

$$C \ominus P = \prod_{i=1}^{m} (C_i \ominus P_\prec) \tag{138}$$

From $C \sqsubseteq P_\prec$ (136), the fact that $P_\prec$ is the lowest conjunct of $P_\prec$ (sic!) and the definition of $\ominus$ it follows that:

$$C \ominus P_\prec = \prod_{i=1}^{m} (C_i \ominus P_\prec)$$

From this and (138) it can be concluded that:

$$C \ominus P = C \ominus P_\prec$$

This implies (133).

**Case 3:** $C = \exists R.C_1$ with a concept $C_1$ and $R \in \mathcal{R}_\mathcal{T}$. Then:
From $P \not\equiv \top$ (135) it follows that:

$$P \neq \top$$

From this, $C = \exists R.C_1$, $C \sqsubseteq P$ (134), $u \notin Roles(P)$ and Lemma 10 it follows that:

$$P = \prod_{i=1}^{n} \exists R.P_i'$$

where $P_i'$ are concepts. Assume w.l.o.g. that:

$$P_\prec = \exists R.P_1' \tag{139}$$

From this, $C \sqsubseteq P$ (134) and the definition of $\ominus$ (Definition 24) it follows that:

$$\exists R.C_1 \ominus P = f_{contract}(\exists R.(C_1 \ominus P_1')) \tag{140}$$

From $C \sqsubseteq P_\prec$ (136), $P_\prec = \exists R.P_1'$ (139) and the definition of $\ominus$ it follows that:

$$C \ominus P_\prec = f_{contract}(\exists R.(C_1 \ominus P_1'))$$

From this and (140) it can be concluded that:

$$C \ominus P = C \ominus P_\prec$$

This proves (133). □

## 7.5. Conclusion

In this chapter, an operator for retracting knowledge from $\mathcal{EL}$ concepts was designed. For this, the AGM postulates for belief set contraction and the postulates for belief base contraction were examined. These postulates were used as a guidance to define properties an $\mathcal{EL}$ knowledge retraction operator should have. Additional properties were developed to make up for the loss of the recovery and relevance postulates which ensure a conservative removal of knowledge. Following this, it was shown that the commutability of subtrahends would conflict with these properties. In a next step, a formal definition for the operator was given. It was then shown that the operator can be used for a fine-grained removal of knowledge from concepts. Moreover, it was also proven that the operator fulfills all the properties from Section 6.2.

The goal of this thesis, the development of an operator for the fine-grained retraction of knowledge from $\mathcal{EL}$ concepts, has therefore been achieved. In the next chapter, a plugin for the ontology editor Protégé is presented that builds upon these results.

# Part IV.
# Implementation

This chapter introduces a plugin for the ontology editor Protégé[55] that can be used to retract knowledge from $\mathcal{EL}$ concepts. The first section explains the reasons behind the choice of Protégé. The following section then gives an overview of the implementation details of the plugin, including its structure and the employed algorithms that were derived from the definition of the operator.

## 8. Reasons for Protégé

Protégé [137] is a free and open-source ontology editor that provides full support for OWL 2 ontologies. It is written in Java and is available under a license alike to the FreeBSD License.[56] Protégé can utilize Description Logic reasoners such as HermiT [111] and Pellet [63] in a transparent way. This makes it possible to compute the hierarchy of classes in the ontology, including inferred subclass relationships, which can then be used to detect modeling errors. In a study conducted by Cardoso [138] in 2007, Protégé was the most used ontology editor among the participants.

From a developer's point-of-view, Protégé's architecture allows programmers to easily develop plugins that can be integrated into the IDE in various ways by using tabs, widgets and menu items. Moreover, new plugins can be combined with existing components, leading to a seamless integration of new features and facilitating the reuse of existing components such as Protégé's *class hierarchy viewer*. Until today, a variety of plugins for Protégé has been developed [126, 139, 140, 141, 142].

For these reasons, Protégé was chosen as the base for the implementation of a plugin that realizes the findings of the previous chapter. For more in-depth information about Protégé, the reader is referred to Musen [137] and the Protégé Wiki[57].

## 9. Implementation

This section describes the implementation of the operator. First, the general structure of the plugin is explained. After this, an algorithm for retracting knowledge from concepts is described which is based on the formal definition of the operator.

[55] See https://protege.stanford.edu/.
[56] See https://github.com/protegeproject/protege/blob/master/license.txt.
[57] See https://protegewiki.stanford.edu/wiki/Main_Page.

## 9.1. Structure

This section gives an overview of the structure of the plugin. It is added to Protégé as an additional tab. This tab consists of three components as displayed in Figure 2. Two of them are regular Protégé components: the *class hierarchy viewer* and the *class*

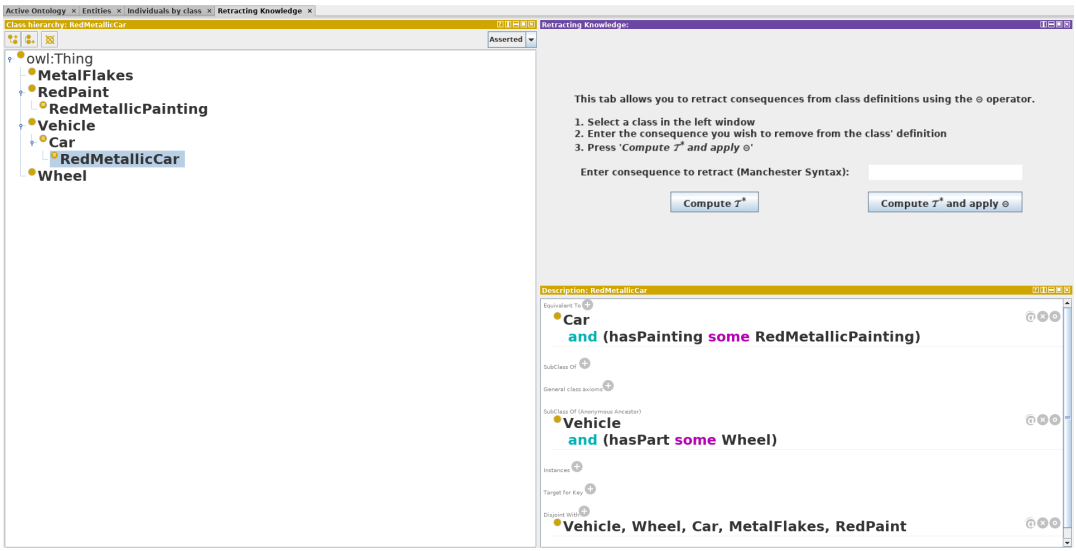


Figure 2: The plugin tab.

*description viewer*. The class hierarchy viewer is located in the left side of Figure 2 and displays the hierarchy of the concepts that are stored in the ontology. In the example case, it shows that the concept $RedMetallicCar$ is subsumed by the concepts $Car$ and $Vehicle$. The class hierarchy viewer is also used to select a class. The information about the selected class is then also available to other components. In the example, the class $RedMetallicCar$ is selected. The class description viewer displays more detailed information about a class and is located in the right lower half of Figure 2. It retrieves the current selected class from the class hierarchy viewer and shows, amongst other things, its definition. In the current example, the class description viewer displays the definition of $RedMetallicCar$. For this, a special syntax for describing OWL 2 concepts is used that is called *Manchester Syntax*.[58] The description of $RedMetallicCar$ corresponds to the following definition:

$$RedMetallicCar \equiv Car \sqcap \exists hasPainting.RedMetallicPainting$$

The third component in the right upper half of Figure 2 is a new addition. It is a component for retracting knowledge from concepts. Figure 3 provides a more detailed view of this component. It can be used to compute the complete expansion of concepts using the button labeled *Compute* $\mathcal{T}^*$. If this button is clicked, the plugin

[58] See https://www.w3.org/TR/owl2-manchester-syntax/.

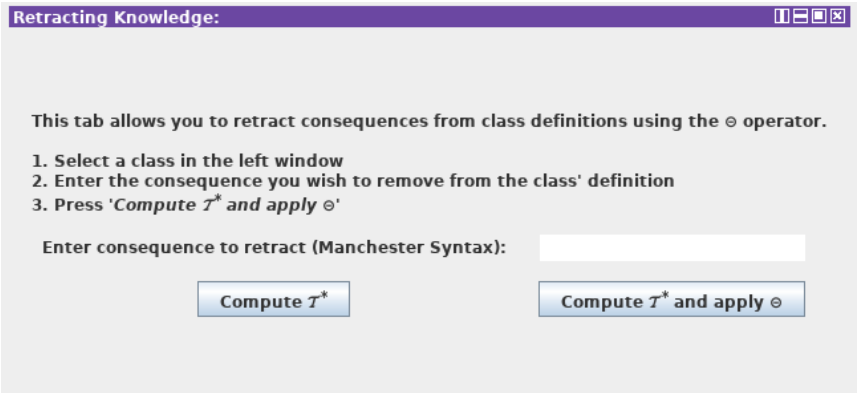


Figure 3: The component for retracting knowledge from concepts.

executes the algorithm *computeCompleteExpansion* which is introduced in the next section. To apply the retraction operation to the currently selected concept, one can click the button labeled *Compute* $\mathcal{T}^*$ *and apply* ⊖. The consequence one wants to retract must be entered into the text field using Manchester Syntax. The plugin then uses Protégé's *OWLExpressionChecker* to perform a highlighting of keywords and a syntax check on the text that is entered. This is shown in Figure 4. If the entered text

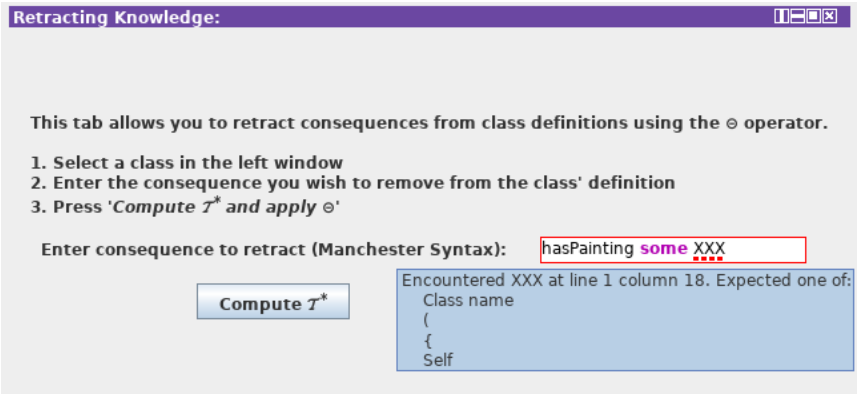


Figure 4: Highlighting keywords and checking the syntax of concept descriptions is performed automatically.

does not correspond to a valid OWL concept, a red error marker is displayed and helpful information to resolve the issue is shown. A click on *Compute* $\mathcal{T}^*$ *and apply* ⊖ then executes the algorithm *retractKnowledge* and passes the currently selected concept and the subtrahend to it. The employed algorithms are explained in the following section.

## 9.2. Algorithms

In this section, the algorithms that are used by the plugin are explained. They were derived from the findings of Part III. In the following, it is assumed that all assumptions from Section 5.2 are fulfilled.

Let $role$ be a function that takes a role restriction and returns the role name and let $filler$ be a function that takes a role restriction and returns the filler. In addition, let $chooseOne$ be a function that takes a set and returns an arbitrary element of it. The algorithm *lookup* is then used to look-up a concept's definition within the TBox.

---

**Algorithm** *lookup*

---

**Input:** $C$, a concept and $\mathcal{T}$, a TBox
**Output:** $C_{def}$, the right-hand side of the definition of $C$ within $\mathcal{T}$ if existent, or else the unchanged concept

1: **if** $C$ is not a named concept in $\mathcal{T}$ **then**
2: $\quad C_{def} \leftarrow C$
3: **else**
4: $\quad C_{def} \leftarrow$ the right-hand side of the definition of $C$ within $\mathcal{T}$
5: **end if**
6: **return** $C_{def}$

---

The algorithm *computeCanonicalExtension* computes the canonical extension of a concept.[59]

---

**Algorithm** *computeCanonicalExtension*

---

**Input:** $C$, a concept and $\mathcal{T}$, a TBox
**Output:** $C'$, the canonical extension of $C$

1: $Count \leftarrow 0$
2: **for all** trivial conjuncts $D$ of $C$ **do**
3: $\quad Count \leftarrow Count + 1$
4: $\quad$ **if** $Count > 1$ **then**
5: $\quad\quad C' \leftarrow C' \sqcap lookup(D, \mathcal{T})$
6: $\quad$ **else**
7: $\quad\quad C' \leftarrow lookup(D, \mathcal{T})$
8: $\quad$ **end if**
9: **end for**
10: **return** $C'$

---

The algorithm *computeCompleteExpansion* computes the complete expansion of a concept.[60] For this, it repeatedly computes the canonical extension of the concept

[59] See Definition 13 in Section 3.3.3.
[60] See Definition 14 in Section 3.3.3.

until the result does not change anymore. At this point, $C^*$ contains the complete expansion of $C$.

**Algorithm** *computeCompleteExpansion*

**Input:** $C$, a concept and $\mathcal{T}$, an acyclic TBox
**Output:** $C^*$, the complete expansion of $C$

```
1: C' ← C
2: repeat
3:    C* ← C'
4:    C' ← computeCanonicalExtension(C', T)
5: until C* = C'
6: return C*
```

The algorithm *getLowestConjunct* computes the lowest conjunct of a concept where an alphabetical ordering of concept names and role names was chosen as a starting point. For an atomic concept or a role restriction this computation is trivial. For a concept that is a non-trivial conjunction all conjuncts are inspected to find the lowest one.

**Algorithm** *getLowestConjunct*

**Input:** $C_{norm}$, a normalized concept
**Output:** the lowest conjunct of $C_{norm}$

```
1: if C_norm is an atomic concept or a role restriction then
2:    return C_norm
3: else  //C_norm must be a non-trivial conjunction
4:    lowest ← ⊤   //⊤ is the largest concept
5:    for all trivial conjuncts D of C_norm do
6:       if isLower(D, lowest) then
7:          lowest ← D
8:       end if
9:    end for
10:   return lowest
11: end if
```

The algorithm *normalize* computes the normal form of a concept $C$. An atomic concept is always in its normal form (line 1 – 2). If $C$ is a role restriction, the operation is pushed into the filler of the role restriction (line 26 – 28). If $C$ is a non-trivial conjunction, then it is tested if conjuncts within $C$ subsume each other. For this, all conjuncts are compared to each other (line 4 – 18). If one conjunct is found that is implied by another conjunct, it is added to the set $S'$ (line 14 – 15). If it happens that a conjunct $D_1$ is already in this set, checking if this conjunct implies other conjuncts can be skipped (line 7–8): all conjuncts that are implied by $D_1$ are already in the set $S'$, because they are also implied by the concept that caused the inclusion of $D_1$ in

$S'$. After the loop completed, all conjuncts that are in the set $S'$ are discarded (line 19) and the remaining conjuncts are normalized (line 20 – 25). The resulting concept of this procedure fulfills the definition of a normalized concept.[61]

**Algorithm** *normalize*

**Input:** $C$, a concept
**Output:** $C_{norm}$, the normal form of $C$

1: **if** $C$ is an atomic concept **then**
2: $C_{norm} \leftarrow C$
3: **else if** $C$ is a non-trivial conjunction **then**
4: $S \leftarrow$ all trivial conjuncts of $C$
5: $S' \leftarrow \emptyset$
6: **for all** $D_1 \in S$ **do**
7: **if** $D_1 \in S'$ **then**
8: **continue**
9: **end if**
10: **for all** $D_2 \in S$ **do**
11: **if** $D_1 = D_2$ **then**
12: **continue**
13: **end if**
14: **if** $D_1 \sqsubseteq D_2$ **then** //Find redundant concepts
15: $S' \leftarrow S' \cup \{D_2\}$
16: **end if**
17: **end for**
18: **end for**
19: $S \leftarrow S - S'$ //Discard all conjuncts that subsume other conjuncts
20: $D \leftarrow chooseOne(S)$
21: $C_{norm} \leftarrow normalize(D)$ //Terminates, because D is a trivial conjunction
22: $S \leftarrow S - \{D\}$
23: **for all** $D \in S$ **do**
24: $C_{norm} \leftarrow C_{norm} +$ "$\sqcap$" $+ normalize(D)$
25: **end for**
26: **else** //$C$ is a role restriction
27: $NewFiller \leftarrow normalize(filler(C))$
28: $C_{norm} \leftarrow$ "$\exists$" $+ role(C) +$ "." $+ NewFiller$
29: **end if**
30: **return** $C_{norm}$

The algorithm *isLower* determines if a concept is lower than another one w.r.t. the order $\prec$.[62] It closely follows the definition of $\prec$.

[61] See Definition 16 in Section 5.1.
[62] See Definition 17 in Section 5.1.

**Algorithm** *isLower* (Part 1)

**Input:** $C_{norm}$ and $P_{norm}$, normalized concepts
**Output:** true iff $C_{norm} \prec P_{norm}$

1: **if** $C_{norm}$ is an atomic concept **then**
2:  **if** $P_{norm}$ is an atomic concept **then**
3:  **if** $P_{norm} = \top$ **then**
4:  **return** true
5:  **else if** $C_{norm} = \top$ **then**
6:  **return** false
7:  **end if**
8:  **return** $isLowerAlphabeticOrder(C_{norm}, P_{norm})$
9:  **else if** $P_{norm}$ is a role restriction **then**
10:  **return** false
11:  **else** $//P_{norm}$ must be a non-trivial conjunction
12:  $P_{\prec} \leftarrow getLowestConjunct(P_{norm})$
13:  **if** $C_{norm} = P_{\prec}$ **then**
14:  **return** true
15:  **end if**
16:  **return** $isLower(C_{norm}, P_{\prec})$
17:  **end if**
18: **else if** $C_{norm}$ is a non-trivial conjunction **then**
19:  $C_{\prec} \leftarrow getLowestConjunct(C_{norm})$
20:  **if** $P_{norm}$ is a non-trivial conjunction **then**
21:  $P_{\prec} \leftarrow getLowestConjunct(P_{norm})$
22:  **if** $C_{\prec} = P_{\prec}$ **then**
23:  $C' \leftarrow C_{norm}$ without the conjunct $C_{\prec}$
24:  $P' \leftarrow P_{norm}$ without the conjunct $P_{\prec}$
25:  **return** $isLower(C', P')$
26:  **else**
27:  **return** $isLower(C_{\prec}, P_{\prec})$
28:  **end if**
29:  **else** $//P_{norm}$ must be a role restriction or an atomic concept
30:  **if** $C_{\prec} = P_{norm}$ **then**
31:  **return** false
32:  **end if**
33:  **return** $isLower(C_{\prec}, P_{norm})$
34:  **end if**

**Algorithm** *isLower* (Part 2)

35: **else** //$C_{norm}$ must be a role restriction
36: **if** $P_{norm}$ is an atomic concept **then**
37: **return** true
38: **else if** $P_{norm}$ is a non-trivial conjunction **then**
39: $P_{\prec} \leftarrow getLowestConjunct(P_{norm})$
40: **if** $C_{norm} = P_{\prec}$ **then**
41: **return** true
42: **end if**
43: **return** $isLower(C_{norm}, P_{\prec})$
44: **else** //$P_{norm}$ must be a role restriction
45: **if** $role(C_{norm}) = role(P_{norm})$ **then**
46: **return** $isLower(filler(C_{norm}), filler(P_{norm}))$
47: **else if** $role(P_{norm})$ is the universal role **then**
48: **return** true
49: **else if** $role(C_{norm})$ is the universal role **then**
50: **return** false
51: **else**
52: **return** $isLowerAlphabeticOrder(role(C_{norm}), role(P_{norm}))$
53: **end if**
54: **end if**
55: **end if**

The algorithm *retractKnowledge* is a wrapper for $retractKnowledgeEx$. It accepts concepts that have not been normalized or which are not completely expanded and applies the necessary transformations before calling $retractKnowledgeEx$.

**Algorithm** *retractKnowledge*

**Input:** $C$, a concept and $P$, a concept that does not contain the universal role and $\mathcal{T}$, an acyclic TBox
**Output:** $C \ominus P$, the result of retracting $P$ from $C$

1: $C^* \leftarrow computeCompleteExpansion(C, \mathcal{T})$
2: $C^*_{norm} \leftarrow normalize(C^*)$
3: $P_{norm} \leftarrow normalize(P)$
4: $Result \leftarrow retractKnowledgeEx(C^*_{norm}, P_{norm})$
5: **return** $Result$

The algorithm *retractKnowledgeEx* computes the result of the retraction of a concept $P_{norm}$ from a concept $C^*_{norm}$. Both concepts must be normalized and $C^*_{norm}$ must be completely expanded. The structure of the algorithm follows immediately from the definition of the operator.[63] Note again that if $C^*_{norm}$ is a role restriction

[63] See Definition 24 in Section 7.1.

such that $C^*_{norm} \sqsubseteq P_{norm}$ and $P_{norm} \neq \top$, then from the normalization of $P_{norm}$, the exclusion of the universal role from $P_{norm}$ and Lemma 10 (Section 7.4) it follows that the lowest conjunct $P_{\prec}$ of $P_{norm}$ is also a role restriction and has the same role as $C^*_{norm}$.

**Algorithm** *retractKnowledgeEx*

**Input:** $C^*_{norm}$, a normalized and completely expanded concept and $P_{norm}$, a normalized concept that does not contain the universal role
**Output:** the normalized result of retracting $P_{norm}$ from $C^*_{norm}$

**if** $C^*_{norm} \not\sqsubseteq P_{norm}$ **then**
  **return** $C^*_{norm}$
**end if**
**if** $C^*_{norm}$ is an atomic concept **then**
  **return** $\top$
**else if** $C^*_{norm}$ is a non-trivial conjunction **then**
  $Result \leftarrow \top$ //Simplifies the generation of the result
  $P_{\prec} \leftarrow getLowestConjunct(P_{norm})$
  **for all** trivial conjuncts $D$ of $C^*_{norm}$ **do**
    $Result \leftarrow Result + \text{“}\sqcap\text{“} + retractKnowledgeEx(D, P_{\prec})$
  **end for**
  **return** $normalize(Result)$
**else** //$C^*_{norm}$ is a role restriction
  **if** $P_{norm} = \top$ **then**
    **return** $\top$
  **end if**
  $P_{\prec} \leftarrow getLowestConjunct(P_{norm})$ //$P_{norm}$ is a role restriction with the same role as $C^*_{norm}$
  $NewFiller \leftarrow retractKnowledgeEx(filler(C^*_{norm}), filler(P_{\prec}))$
  **if** $NewFiller \equiv \top$ **then** //Replaces $f_{contract}$
    **return** $\top$
  **else**
    $Result \leftarrow \text{“}\exists\text{“} + role(C^*_{norm}) + \text{“.“} + NewFiller$
    **return** $normalize(Result)$
  **end if**
**end if**

In this section the algorithms that are used by the plugin were explained. In the next chapter a final conclusion is drawn and an outlook into future work is given.

# Part V.
# Conclusion and Future Work

In this chapter the findings of the thesis are summarized and a final conclusion is drawn. The chapter then closes with an outlook into possible future work.

## 10. Conclusion

In this work, a short introduction to ontologies and OWL, one of the major ontology languages, was given.[64] The problem of ontology change was introduced and it was argued how a formal operator for the retraction of knowledge can support this process. It was then explained why the Description Logic $\mathcal{EL}$ was chosen as a starting point in the development of this operator. Following this, an introduction to Description Logic was given and the logic $\mathcal{EL}$ was formally introduced. In a next step, related work was examined to find approaches that could provide a guideline for the development of the operator. This included research that is concerned with computing the difference between concepts as well as methods for debugging Description Logic knowledge bases. It was found that these approaches cannot offhandedly be transferred to $\mathcal{EL}$ or that they work in a too coarse-grained way and thus remove too much knowledge from concepts. In addition, the AGM postulates for belief set contraction and the postulates for belief base contraction were examined in order to judge their applicability in the development of the operator. As a negative result, it was found that the recovery and relevance postulates are not naïvely applicable to this task. Based on these findings, general requirements for an $\mathcal{EL}$ knowledge retraction operator were stated and formal properties were derived from the postulates of belief base and belief set contraction. In addition, properties were formulated which, amongst others, made up for the non-applicability of the recovery and relevance postulates and the accompanied loss of mechanisms that ensure a minimality of knowledge removal during retraction. Based on these requirements, the operator $\ominus$ was defined and it was formally proven that the operator fulfills the specified properties. It was further shown that the operator is applicable to the task of a fine-grained removal of knowledge from $\mathcal{EL}$ concepts. In addition, an example was given that showed how the operator can be combined with laconic justifications to possibly assist a human ontology editor in the process of ontology change. To complete this work, a plugin for the ontology editor Protégé was developed. For this, pseudo-code algorithms were derived from the formal definition of the operator to provide guidance in the programming of the plugin. Moreover, the plugin was designed in such a way that it integrates well with Protégé's user interface and provides the possibility to enter the subtrahend using Manchester Syntax.

[64] D'Aquin et al. [102] automatically collected more than 25,000 semantic documents from the web in 2007, of which 6200 were OWL ontologies.

The aim of this thesis was the development of a formal operator that facilitates the removal of knowledge from $\mathcal{EL}$ concepts. With the development of $\ominus$, this goal has been achieved. However, there is still room for future research. Possible starting points for this are discussed in the next section.

## 11. Future Work

This section gives an overview of possible starting points for future work. Such work could cover an analysis of alternative solutions to the design decision that were made in the development of the operator. Moreover, future work could focus on extending the plugin or it could be concerned with the development of a recovery operator based on $\ominus$. This is discussed in the following.

**Properties:** The development of the operator required the specification of desirable properties of the retraction operation. This included properties that limit the removal of knowledge since it was found that the recovery and relevance postulates are not naïvely applicable at concept level. For this, the properties independence and informational economy were introduced. Future work could explore other properties that fulfill this purpose and compare them to the current ones.

**Scope:** The Description Logic $\mathcal{EL}$ was used as a starting point in the development of the operator. In a next step, the operator could be extended to $\mathcal{EL}^{++}$ [27, 28]. This extension would provide full support of all OWL 2 EL expressions and would make it possible to apply the operator to large biomedical ontologies such as SNOMED CT.

**Plugin:** Concerning the plugin, future work could extended it in such a way that the order $\prec$ is defined on-the-fly the first time it is needed. It could inform the human ontology editor when a choice in the retraction of knowledge arises and query him to select the concept he likes to give up first, hence incrementally building the order $\prec$ at runtime. In addition, future work could explore the possibility of a combination of the operator with plugins that compute laconic justifications, such as the *Explanation Workbench*.[65] Here the operator could be used to automatically retract those parts from concepts that were identified to cause an unwanted consequence.[66]

**Recovery:** The findings of this work could provide the base for the future development of a recovery operator. Such an operator could make it possible to restore knowledge that was once present in the ontology. To achieve this, one could make use of axiom labels to annotate axioms with recovery information. The usage of axiom labels would also have the benefit that it would be possible to

[65] See http://owl.cs.manchester.ac.uk/research/explanation/.
[66] See again Example 14 in Section 7.3.

store additional meta-knowledge about the retraction operation. This knowledge could include the name of the user who authorized the change, the date of the retraction operation and a human-readable description on why the operation was performed. This meta-knowledge could then be used as an aid in the recovery process and to compute *provenance information* [143] about why certain knowledge is not present anymore.[67] The technical fundamentals to store this knowledge are available in OWL 2 through so-called *annotations*.[68] A naïve approach to a recovery operator could look like the following. Let $\bigoplus$ denote the recovery operator and let $C$ and $P$ denote concepts. Then assume there exists a way to compute recovery information $\mathcal{R}_{C \ominus P}$ to every operation $C \ominus P$, such that:

$$(C \ominus P) \bigoplus \mathcal{R}_{C \ominus P} \equiv C$$

In addition, let $A$ be an axiom and let $label_A$ be the annotation of $A$. Initially, let $label_A = ()$, i.e. $label_A$ is an empty sequence. Let $\frown$ denote sequence concatenation such that for two sequences with $n$ and $m$ elements:

$$(a_1, a_2, \ldots, a_n) \frown (b_1, b_2, \ldots, b_m) = (a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_m)$$

Given this, the process of removing knowledge from an $\mathcal{EL}$ concept can be extended as follows. Every time a consequence is removed from $A$, the corresponding recovery information is computed. This information is then appended to the label of $A$ together with additional meta-knowledge about the operation. This is outlined by the following alternative version of the algorithm *retractKnowledge*.

**Algorithm** *retractKnowledge*

**Input:** $C$ and $P$, concepts and $\mathcal{T}$, an acyclic TBox and $Meta$, meta-knowledge about the operation
**Output:** $C \ominus P$, the result of retracting $P$ from $C$

1: $C^* \leftarrow computeCompleteExpansion(C, \mathcal{T})$
2: $C^*_{norm} \leftarrow normalize(C^*)$
3: $P_{norm} \leftarrow normalize(P)$
4: $Label_C \leftarrow getAxiomLabel(getAxiomOf(C, \mathcal{T}))$
5: $Result \leftarrow retractKnowledgeEx(C^*_{norm}, P_{norm})$
6: $Recovery \leftarrow computeRecoveryInformation(C^*_{norm}, Result)$
7: $Label'_C \leftarrow Label_C \frown ((Recovery, P, Meta))$
8: $storeAxiomLabel(getAxiomOf(C, \mathcal{T}), Label'_C)$
9: **return** $Result$

[67] Schenk et al. [143] make use of axiom labels to attach provenance information to axioms. This information can then be used to debug the ontology and to judge the trustworthiness of inferred knowledge.
[68] See https://www.w3.org/TR/owl2-syntax/#Annotations.

The functions $getAxiomLabel$ and $storeAxiomLabel$ are assumed to obtain and update an axiom label, respectively. The function $computeRecoveryInformation$ is assumed to compute $\mathcal{R}_{C \ominus P}$ given a concept $C$ and the result of $C \ominus P$.
A subsumption relationship $C \sqsubseteq P'$ could then be restored by traversing backwards over the label of the axiom that defines $C$. During this, one could undo all previous knowledge retraction operations until $C \sqsubseteq P'$ is restored. However, at this point $C$ could also contain other restored knowledge that is unrelated to $P'$. This happens in the case that after the retraction of $P'$ another retraction operation was applied to $C$. To remove this knowledge again, one could traverse forward over the label of the axiom and re-apply every retraction operation that does not destroy the subsumption $C \sqsubseteq P'$. This is outlined by the algorithm $restoreKnowledge$.

**Algorithm** *restoreKnowledge*

**Input:** $C$ and $P$, concepts and $\mathcal{T}$, an acyclic TBox
**Output:** $C'$, such that $C' \sqsubseteq P$ or the unchanged concept $C$

1: $Axiom_C \leftarrow getAxiomOf(C, \mathcal{T})$
2: $Label_C \leftarrow getAxiomLabel(Axiom_C)$
3: $n \leftarrow size(Label_C)$ *//Get the number of elements in* $Label_C$
4: $i \leftarrow n - 1$
5: $C' \leftarrow C$
6: **while** $i >= 0$ **do** *//Iterate backwards over the retraction operations*
7: $(Recovery, Operation, Meta) \leftarrow getIthEntry(Label_C, i)$
8: $C' \leftarrow C' \oplus Recovery$ *//Undo the ith retraction*
9: **if** $C' \sqsubseteq P$ **then** *//Test if the consequence* $P$ *has been restored*
10: $i \leftarrow i + 1$
11: $C'' \leftarrow C'$
12: **while** $i < n$ **do***//Redo as many retractions as possible*
13: $(Recovery, Operation, Meta) \leftarrow getIthEntry(Label_C, i)$
14: $C' \leftarrow retractKnowledge(C'', Operation, \mathcal{T}, Meta)$
15: **if** $C' \not\sqsubseteq P$ **then**
16: $C' \leftarrow C''$*//Do not remove the entailment of* $P$ *again*
17: **end if**
18: $C'' \leftarrow C'$
19: $i \leftarrow i + 1$
20: **end while**
21: **return** $C'$
22: **end if**
23: $i \leftarrow i - 1$
24: **end while**
25: **return** $C$

The algorithm returns a concept $C'$ such that $C' \sqsubseteq P$ or the unchanged con-

cept $C$, if it was not possible to restore the subsumption relationship. In the algorithm, the function $size$ is assumed to return the number of elements of a given sequence. Moreover, given an index $i$ and a sequence $S$, the function $getIthEntry$ is assumed to return the $i$th entry of $S$. Note that for reasons of simplicity, the algorithm does not update the axiom. In addition, one would need to provide formal proofs to make sure that the outlined algorithms work as intended.

This section gave an outlook into future work and discussed possible starting points. It further sketched two algorithms that can be used to store meta-knowledge about a retraction operation and to restore knowledge that was once present in an ontology.

## Acknowledgments

Ich möchte mich bei all denjenigen bedanken, die mich während meiner Arbeit unterstützt haben. Mein Dank gebührt zunächst Herrn Professor Dr. Steffen Staab und Frau Dr. Claudia Schon, die es mir ermöglicht haben diese Arbeit am WeST Institut zu verfassen und mir während meiner Arbeit hilfreiche Anregungen und Feedback gaben. Ausdrücklich bedanken möchte ich mich insbesondere bei Frau Dr. Schon, welche stets die Geduld hatte, unzählige Fragen meinerseits zu beantworten und mir hilfreich zur Seite zu stehen. Ohne Ihr breites Wissen im Bereich der Beschreibungslogik wäre diese Arbeit für mich nicht denkbar gewesen.

Danken möchte ich auch meinen Freunden, die mich während der Arbeit immer unterstützt haben, insbesonders meinem Mitbewohner Daniel. Besonderer Dank gebührt auch meiner Freundin Michelle, die mich stets motiviert hat, aber bei der ich auch immer die notwendige Ruhe finden konnte. Danken möchte ich zudem meinen Eltern Gerhard und Judith Heinz, für ihre unbeschränkte Unterstützung und für ihr unbegrenztes Verständnis, wodurch sie mir erst das Studium der Informatik ermöglicht haben. Danken möchte ich auch meinem Bruder Björn für die zahlreichen Fragen, die er mir während meines Studiums beantwortet hat. Ich wünsche Dir weiterhin viel Erfolg bei Deiner Forschung an einem Magnonen-Computer! Es ist toll, von solchen Menschen umgeben zu sein.

## References


[1] G. Flouris, D. Manakanatas *et al.*, "Ontology Change: Classification and Survey," *The Knowledge Engineering Review*, vol. 23, no. 2, pp. 117 – 152, 2008.

[2] R. Cornet, M. Nyström, and D. Karlsson, "User-Directed Coordination in SNOMED CT," in *Proceedings of the 14th World Congress on Medical and Health Informatics (MEDINFO'2013), Copenhagen, Denmark*, ser. Studies in Health Technology and Informatics, C. U. Lehmann, E. Ammenwerth, and C. Nøhr, Eds., vol. 192. Amsterdam, The Netherlands: IOS Press, 2013, pp. 72 – 76.

[3] S. O. Hansson, "Belief Contraction Without Recovery," *Studia Logica*, vol. 50, no. 2, pp. 251 – 260, 1991.

[4] ——, *A Textbook of Belief Dynamics: Theory Change and Database Updating*, 1st ed., ser. Applied Logic Series. Dordrecht, Netherlands: Springer Netherlands, 1999, vol. 11.

[5] C. E. Alchourrón, P. Gärdenfors, and D. Makinson, "On the Logic of Theory Change: Partial Meet Contraction and Revision Functions," *The Journal of Symbolic Logic*, vol. 50, no. 2, pp. 510 – 530, 1985.

[6] F. M. Suchanek, C. Menard *et al.*, "Can You Imagine... A Language for Combinatorial Creativity?" in *First Part of the Proceededings of the 15th International Semantic Web Conference (ISWC'2016), Kōbe, Japan*, ser. Lecture Notes in Computer Science, P. T. Groth, E. Simperl *et al.*, Eds., vol. 9981. Cham, Switzerland: Springer International Publishing, 2016, pp. 532 – 548.

[7] M. Horridge, B. Parsia, and U. Sattler, "Laconic and Precise Justifications in OWL," in *Proceedings of the Seventh International Semantic Web Conference (ISWC'2008), Karlsruhe, Germany*, ser. Lecture Notes in Computer Science, A. P. Sheth, S. Staab *et al.*, Eds., vol. 5318. Berlin & Heidelberg, Germany: Springer-Verlag Berlin Heidelberg, 2008, pp. 323 – 338.

[8] R. Studer, V. R. Benjamins, and D. Fensel, "Knowledge Engineering: Principles and Methods," *Data & Knowledge Engineering*, vol. 25, no. 1-2, pp. 161 – 197, 1998.

[9] T. R. Gruber, "A Translation Approach to Portable Ontology Specifications," *Knowledge Acquisition*, vol. 5, no. 2, pp. 199 – 220, 1993.

[10] W. N. Borst, "Construction of Engineering Ontologies for Knowledge Sharing and Reuse," Ph.D. dissertation, University of Twente, Enschede, The Netherlands, 1997.

[11] N. Shadbolt, T. Berners-Lee, and W. Hall, "The Semantic Web Revisited," *IEEE Intelligent Systems*, vol. 21, no. 3, pp. 96 – 101, 2006.

[12] R. A. Cote and S. Robboy, "Progress in Medical Information Management: Systematized Nomenclature of Medicine (SNOMED)," *Journal of the American Medical Association*, vol. 243, no. 8, pp. 756 – 762, 1980.

[13] A. L. Rector, J. Rogers *et al.*, "OpenGALEN: Open Source Medical Terminology and Tools," in *American Medical Informatics Association Annual Symposium (AMIA'2003), Washington, District of Columbia, USA*, M. A. Musen, C. P. Friedman, and J. M. Teich, Eds. Bethesda, Maryland, USA: American Medical Informatics Association, 2003, p. 982.

[14] M. Ashburner, C. A. Ball *et al.*, "Gene Ontology: Tool for the Unification of Biology," *Nature Genetics*, vol. 25 - 29, no. 1, p. 25, 2000.

[15] Y. Tzitzikas, C. Allocca *et al.*, "Integrating Heterogeneous and Distributed Information About Marine Species Through a Top Level Ontology," in *Proceedings of the Seventh Conference on Metadata and Semantics Research (MTSR'2013), Thessaloniki, Greece*, ser. Communications in Computer and Information Science, E. Garoufallou and J. Greenberg, Eds., vol. 390. Cham, Switzerland: Springer International Publishing, 2013, pp. 289 – 301.

[16] C. J. Mungall, C. Torniai *et al.*, "Uberon, an Integrative Multi-Species Anatomy Ontology," *Genome Biology*, vol. 13, no. 1, p. R5, 2012.

[17] M. Hepp, "Goodrelations: An Ontology for Describing Products and Services Offers on the Web," in *Proceedings of the 16th International Conference on Knowledge Engineering (EKAW'2008), Acitrezza, Italy*, ser. Lecture Notes in Computer Science, A. Gangemi and J. Euzenat, Eds., vol. 5268. Berlin & Heidelberg, Germany: Springer-Verlag Berlin Heidelberg, 2008, pp. 329 – 346.

[18] D. Brickley and L. Miller, "FOAF Vocabulary Specification 0.99," 2014, [Online]. Available: http://xmlns.com/foaf/spec/, accessed: 2018-01-07.

[19] SNOMED International, "What is SNOMED CT?" 2018, [Online]. Available: https://www.snomed.org/snomed-ct/what-is-snomed-ct, accessed: 2018-01-14.

[20] ——, "SNOMED International Edition," vol. 20180131, 2018, [Online]. Available: http://browser.ihtsdotools.org, accessed: 2018-02-19.

[21] T. G. O. Consortium, "Expansion of the Gene Ontology Knowledgebase and Resources," *Nucleic Acids Research*, vol. 45 (Database-Issue), pp. D331 – D338, 2017.

[22] B. Smith, M. Ashburner *et al.*, "The OBO Foundry: Coordinated Evolution of Ontologies to Support Biomedical Data Integration," *Nature Biotechnology*, vol. 25, no. 11, pp. 1251 – 1255, 2007.

[23] Gene Ontology Consortium, "About The Gene Ontology Project," 2018, [Online]. Available: http://www.geneontology.org/page/about, accessed: 2018-03-17.

[24] ——, "AmiGO 2 Live Search," 2018, [Online]. Available: http://amigo.geneontology.org/amigo/search/annotation, accessed: 2018-03-17.

[25] The OWL Working Group, "OWL 2 Web Ontology Language Document Overview (Second Edition)," *W3C Recommendation,* World Wide Web Consortium (W3C), 2012, [Online]. Available: https://www.w3.org/TR/2012/REC-owl2-overview-20121211/, accessed: 2017-12-12.

[26] ——, "OWL 2 Web Ontology Language Profiles (Second Edition)," *W3C Recommendation,* World Wide Web Consortium (W3C), 2012, [Online]. Available: https://www.w3.org/TR/owl2-profiles, accessed: 2018-01-02.

[27] F. Baader, S. Brandt, and C. Lutz, "Pushing the $\mathcal{EL}$ Envelope," in *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI'05), Edinburgh, Scotland, UK*, L. P. Kaelbling and A. Saffiotti, Eds. Denver, Colorado, USA: Professional Book Center, 2005, pp. 364 – 369.

[28] F. Baader, C. Lutz, and S. Brandt, "Pushing the $\mathcal{EL}$ Envelope Further," in *Proceedings of the Fourth Workshop on OWL: Experiences and Directions (OWLED'2008DC), Co-Located with the Second W3C OWL Working Group Face-to-Face Meeting, Washington, District of Columbia, USA*, ser. CEUR Workshop Proceedings, K. Clark and P. F. Patel-Schneider, Eds., vol. 496. RWTH Aachen, Aachen, Germany: CEUR-WS.org, 2008. [Online]. Available: http://ceur-ws.org/Vol-496

[29] S. Brandt, "Polynomial Time Reasoning in a Description Logic with Existential Restrictions, GCI Axioms, and - What Else?" in *Proceedings of the 16th Eureopean Conference on Artificial Intelligence (ECAI'2004), Co-Located with the Prestigious Applicants of Intelligent Systems (PAIS'2004), Valencia, Spain*, R. L. de Mántaras and L. Saitta, Eds. Amsterdam, The Netherlands: IOS Press, 2004, pp. 298 – 302.

[30] F. Baader, G. Brewka, and O. F. Gil, "Adding Threshold Concepts to the Description Logic $\mathcal{EL}$," in *10th International Symposium on Frontiers of Combining Systems (FroCoS'2015), Wroclaw, Poland*, ser. Lecture Notes in Computer Science, C. Lutz and S. Ranise, Eds., vol. 9322. Cham, Switzerland: Springer International Publishing, 2015, pp. 33 – 48.

[31] V. Gutiérrez-Basulto, J. C. Jung, and T. Schneider, "Lightweight Description Logics and Branching Time: A Troublesome Marriage," in *Proceedings of the*

*14th International Conference on Principles of Knowledge Representation and Reasoning (KR'2014), Vienna, Austria*, C. Baral, G. D. Giacomo, and T. Eiter, Eds. Menlo Park, California, USA: AAAI Press, 2014.

[32] S. Borgwardt and V. Thost, "Temporal Query Answering in the Description Logic $\mathcal{EL}$," in *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI'2015), Buenos Aires, Argentina*, Q. Yang and M. Wooldridge, Eds. Menlo Park, California, USA: AAAI Press, 2015, pp. 2819 – 2825.

[33] C. Lutz, D. Toman, and F. Wolter, "Conjunctive Query Answering in the Description Logic $\mathcal{EL}$ Using a Relational Database System," in *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI'2009), Pasadena, California, USA*, C. Boutilier, Ed. Menlo Park, California, USA: AAAI Press, 2009, pp. 2070 – 2075.

[34] P. Hansen, C. Lutz *et al.*, "Efficient Query Rewriting in the Description Logic $\mathcal{EL}$ and Beyond," in *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI'2015), Buenos Aires, Argentina*, Q. Yang and M. Wooldridge, Eds. Menlo Park, California, USA: AAAI Press, 2015, pp. 3034 – 3040.

[35] T. Berners-Lee, J. Hendler *et al.*, "The Semantic Web," *Scientific American*, vol. 284, no. 5, pp. 28 – 37, 2001.

[36] F. Zablith, G. Antoniou *et al.*, "Ontology Evolution: A Process-Centric Survey," *Knowledge Engineering Review*, vol. 30, no. 1, pp. 45 – 75, 2015.

[37] E. P. B. Simperl, "Reusing Ontologies on the Semantic Web: A Feasibility Study," *Data & Knowledge Engineering*, vol. 68, no. 10, pp. 905 – 925, 2009.

[38] C. Ochs, Y. Perl *et al.*, "An Empirical Analysis of Ontology Reuse in BioPortal," *Journal of Biomedical Informatics*, vol. 71, pp. 165 – 177, 2017.

[39] P. L. Whetzel, N. F. Noy *et al.*, "BioPortal: Enhanced Functionality via New Web Services from the National Center for Biomedical Ontology to Access and Use Ontologies in Software Applications," *Nucleic Acids Research*, vol. 39 (Web Server Issue), pp. 541 – 545, 2011.

[40] The Board of Trustees of Leland Stanford Junior University, "NCBO BioPortal," 2018, [Online]. Available: https://bioportal.bioontology.org, accessed: 2018-03-04.

[41] J. Banerjee, W. Kim *et al.*, "Semantics and Implementation of Schema Evolution in Object-Oriented Databases," in *Proceedings of the 1987 ACM Special Interest Group on Management of Data Annual Conference (SIGMOD'87), San Francisco, California, USA*, U. Dayal and I. L. Traiger, Eds. New York, New York, USA: ACM, 1987, pp. 311 – 322.

[42] J. F. Roddick, "Schema Evolution in Database Systems - An Annotated Bibliography," *SIGMOD Record*, vol. 21, no. 4, pp. 35 – 40, 1992.

[43] E. Rahm and P. A. Bernstein, "A Survey of Approaches to Automatic Schema Matching," *International Journal on Very Large Databases*, vol. 10, no. 4, pp. 334 – 350, 2001.

[44] ——, "An Online Bibliography on Schema Evolution," *SIGMOD Record*, vol. 35, no. 4, pp. 30 – 31, 2006.

[45] N. F. Noy and M. Klein, "Ontology Evolution: Not the Same as Schema Evolution," *Knowledge and Information Systems*, vol. 6, no. 4, pp. 428 – 440, 2004.

[46] M. C. A. Klein, "Change Management for Distributed Ontologies," Ph.D. dissertation, Free University of Amsterdam, Amsterdam, The Netherlands, 2004.

[47] H. Kondylakis, G. Flouris, and D. Plexousakis, "Ontology and Schema Evolution in Data Integration: Review and Assessment," in *Second Part of the Proceedings of the Confederated International Conferences CoopIS, DOA, IS, and ODBASE: On the Move to Meaningful Internet Systems (OTM'2009), Vilamoura, Portugal*, ser. Lecture Notes in Computer Science, R. Meersman, T. S. Dillon, and P. Herrero, Eds., vol. 5871. Berlin & Heidelberg, Germany: Springer-Verlag Berlin Heidelberg, 2009, pp. 932 – 947.

[48] P. Haase, F. van Harmelen *et al.*, "A Framework for Handling Inconsistency in Changing Ontologies," in *Proceedings of the Fourth International Semantic Web Conference (ISWC'2005), Galway, Ireland*, ser. Lecture Notes in Computer Science, Y. Gil, E. Motta *et al.*, Eds., vol. 3729. Berlin & Heidelberg, Germany: Springer-Verlag Berlin Heidelberg, 2005, pp. 353 – 367.

[49] A. L. Rector, N. Drummond *et al.*, "OWL Pizzas: Practical Experience of Teaching OWL-DL: Common Errors & Common Patterns," in *Proceedings of the 14th International Conference on Engineering Knowledge in the Age of the Semantic Web (EKAW'2004), Whittlebury Hall, UK*, ser. Lecture Notes in Computer Science, E. Motta, N. Shadbolt *et al.*, Eds., vol. 3257. Berlin & Heidelberg, Germany: Springer-Verlag Berlin Heidelberg, 2004, pp. 63 – 81.

[50] M. R. Quillian, "Word Concepts: A Theory and Simulation of Some Basic Semantic Capabilities," *Systems Research and Behavioral Science*, vol. 12, no. 5, pp. 410 – 430, 1967.

[51] M. Minsky, *A Framework for Representing Knowledge*, 1st ed., ser. Computer Science Series. New York, New York, USA: McGraw-Hill, 1975.

[52] R. J. Brachman, "A Structural Paradigm for Representing Knowledge," Ph.D. dissertation, Harvard University, Cambridge, Massachusetts, USA, 1977.

[53] R. J. Brachman and J. G. Schmolze, "An Overview of the KL-ONE Knowledge Representation System," *Cognitive Science*, vol. 9, no. 2, pp. 171 – 216, 1985.

[54] R. MacGregor and R. Bates, "The Loom Knowledge Representation Language," Ph.D. dissertation, Marina Del Rey Information Sciences Institute, University of Southern California, Los Angeles, California, USA, 1987.

[55] R. M. MacGregor, "Inside the LOOM Description Classifier," *SIGART Bulletin*, vol. 2, no. 3, pp. 88 – 92, 1991.

[56] C. Peltason, "The BACK System - An Overview," *SIGART Bulletin*, vol. 2, no. 3, pp. 114 – 119, 1991.

[57] F. Baader and B. Hollunder, "KRIS: Knowledge Representation and Inference System," *SIGART Bulletin*, vol. 2, no. 3, pp. 8 – 14, 1991.

[58] P. Bresciani, E. Franconi, and S. Tessaris, "Implementing and Testing Expressive Description Logics: Preliminary Report," in *Proceedings of the 1995 International Workshop on Description Logics (DL'95), Rome, Italy*, A. Borgida, M. Lenzerini *et al.*, Eds. Rome, Italy: Dipartimento di Informatica e Sistemistica, Sapienza – Università di Roma, 1995, pp. 131 – 139.

[59] I. Horrocks, "FaCT and iFaCT," in *Proceedings of the 1999 International Workshop on Description Logics (DL'99), Linköping, Sweden*, ser. CEUR Workshop Proceedings, P. Lambrix, A. Borgida *et al.*, Eds., vol. 22. RWTH Aachen, Aachen, Germany: CEUR-WS.org, 1999. [Online]. Available: http://ceur-ws.org/Vol-22

[60] I. Horrocks, U. Sattler, and S. Tobies, "Reasoning with Individuals for the Description Logic $\mathcal{SHIQ}$," in *Proceedings of the 17th International Conference on Automated Deduction (CADE'17), Pittsburgh, Pennsylvania, USA*, ser. Lecture Notes in Computer Science, D. A. McAllester, Ed., vol. 1831. Cham, Switzerland: Springer International Publishing, 2000, pp. 482 – 496.

[61] V. Haarslev and R. Möller, "RACER System Description," in *Proceedings of the First International Joint Conference on Automated Reasoning (IJCAR'2001), Siena, Italy*, ser. Lecture Notes in Computer Science, R. Goré, A. Leitsch, and T. Nipkow, Eds., vol. 2083. Berlin & Heidelberg, Germany: Springer-Verlag Berlin Heidelberg, 2001, pp. 701 – 706.

[62] E. Sirin and B. Parsia, "Pellet: An OWL DL Reasoner," in *Proceedings of the 2004 International Workshop on Description Logics (DL'2004), Whistler, British Columbia, Canada*, ser. CEUR Workshop Proceedings, V. Haarslev and R. Möller, Eds., vol. 104. RWTH Aachen, Aachen, Germany: CEUR-WS.org, 2004. [Online]. Available: http://ceur-ws.org/Vol-104

[63] E. Sirin, B. Parsia *et al.*, "Pellet: A Practical OWL-DL Reasoner," *Journal of Web Semantics*, vol. 5, no. 2, pp. 51 – 53, Jun. 2007.

[64] D. Tsarkov and I. Horrocks, "FaCT++ Description Logic Reasoner: System Description," in *Proceedings of the Third International Joint Conference on Automated Reasoning (IJCAR'2006), Seattle, Washington, USA*, ser. Lecture Notes in Computer Science, U. Furbach and N. Shankar, Eds., vol. 4130. Berlin & Heidelberg, Germany: Springer-Verlag Berlin Heidelberg, 2006, pp. 292 – 297.

[65] P. J. Hayes, "The Logic of Frames," in *Readings in Artificial Intelligence*, 1st ed., B. L. Webber and N. J. Nilsson, Eds. San Francisco, California, USA: Morgan Kaufmann Publishers, 1981, pp. 451 – 458.

[66] W. A. Woods, "What's In a Link: Foundations for Semantic Networks," in *Representation and Understanding*, 1st ed., D. G. Bobrow and A. Collins, Eds. San Francisco, California, USA: Morgan Kaufmann Publishers, 1975, pp. 35 – 82.

[67] R. J. Brachman, "Structured Inheritance Networks," Bolt Beranek and Newman, Cambridge, Massachusetts, USA, Tech. Rep. 3742, 1978.

[68] W. A. Woods and J. G. Schmolze, "The KL-ONE Family," *Computers & Mathematics with Applications*, vol. 23, no. 2-5, pp. 133 – 177, 1992.

[69] R. J. Brachman and H. J. Levesque, "Assertions in KL-One," in *Proceedings of the Second KL-ONE Workshop, Cambridge, Massachusetts, USA*, J. G. Schmolze and R. J. Brachman, Eds. Cambridge, Massachusetts, USA: Bolt Beranek and Newman, 1982, pp. 8 – 17.

[70] R. J. Brachman, R. Fikes, and H. J. Levesque, "KRYPTON: A Functional Approach to Knowledge Representation," *IEEE Computer*, vol. 16, no. 10, pp. 67 – 73, 1983.

[71] P. F. Patel-Schneider, "Undecidability of Subsumption in NIKL," *Artificial Intelligence*, vol. 39, no. 2, pp. 263 – 272, 1989.

[72] B. Nebel, *Reasoning and Revision in Hybrid Representation Systems*, 1st ed., ser. Lecture Notes in Computer Science. Berlin & Heidelberg, Germany: Springer-Verlag Berlin Heidelberg, 1990, vol. 422.

[73] ——, "Terminological Reasoning is Inherently Intractable," *Artificial Intelligence*, vol. 43, no. 2, pp. 235 – 249, 1990.

[74] T. Kaczmarek, R. Bates, and G. Robins, "Recent Developments in NIKL," in *Second Part of the Proceedings of the Fifth National Conference on Artificial Intelligence (AAAI'86), Philadelphia, Pennsylvania, USA*, T. Kehler and S. J. Rosenschein, Eds. San Francisco, California, USA: Morgan Kaufmann Publishers, 1986, pp. 978 – 985.

[75] M. Schmidt-Schauß, "Subsumption in KL-ONE is Undecidable," in *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning (KR'89), Toronto, Canada*, R. J. Brachman, H. J. Levesque, and R. Reiter, Eds. San Francisco, California, USA: Morgan Kaufmann Publishers, 1989, pp. 421 – 431.

[76] A. Borgida, R. J. Brachman *et al.*, "CLASSIC: A Structural Data Model for Objects," in *Proceedings of the 1989 ACM International Conference on Management of Data (SIGMOD'89), Portland, Oregon, USA*, J. Clifford, B. G. Lindsay, and D. Maier, Eds. New York, New York, USA: ACM, 1989, pp. 58 – 67.

[77] P. F. Patel-Schneider, D. L. McGuinness, and A. Borgida, "The CLASSIC Knowledge Representation System: Guiding Principles and Implementation Rationale," *SIGART Bulletin*, vol. 2, no. 3, pp. 108 – 113, 1991.

[78] M. Schmidt-Schauß and G. Smolka, "Attributive Concept Descriptions with Complements," *Artificial Intelligence*, vol. 48, no. 1, pp. 1 – 26, 1991.

[79] P. F. Patel-Schneider and I. Horrocks, "DLP and FaCT," in *Proceedings of the 1999 International Conference on Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX'99), Saratoga Springs, New York, USA*, ser. Lecture Notes in Computer Science, N. V. Murray, Ed., vol. 1617. Berlin & Heidelberg, Germany: Springer-Verlag Berlin Heidelberg, 1999, pp. 19 – 23.

[80] K. Schild, "A Correspondence Theory for Terminological Logics: Preliminary Report," in *Proceedings of the 12th International Joint Conference on Artificial Intelligence (JCAI'91), Sydney, New South Wales, Australia*, J. Mylopoulos and R. Reiter, Eds. San Francisco, California, USA: Morgan Kaufmann Publishers, 1991, pp. 466 – 471.

[81] A. Borgida, "On the Relative Expressiveness of Description Logics and Predicate Logics," *Artificial Intelligence*, vol. 82, no. 1-2, pp. 353 – 367, 1996.

[82] I. Horrocks and P. F. Patel-Schneider, "Optimising Propositional Modal Satisfiability for Description Logic Subsumption," in *Proceedings of the Fourth International Conference on Artificial Intelligence and Symbolic Computation (AISC'98), Plattsburgh, New York, USA*, ser. Lecture Notes in Computer Science, J. Calmet and J. A. Plaza, Eds., vol. 1476. Berlin & Heidelberg, Germany: Springer-Verlag Berlin Heidelberg, 1998, pp. 234 – 246.

[83] D. Koller, A. Y. Levy, and A. Pfeffer, "P-CLASSIC: A Tractable Probablistic Description Logic," in *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI'97) and Ninth Innovative Applications of Artificial Intelligence Conference (IAAI'97), Providence, Rhode Island, USA*, B. Kuipers and B. L. Webber, Eds. Menlo Park, California, USA: AAAI Press, 1997, pp. 390 – 397.

[84] T. Lukasiewicz, "Fuzzy Description Logic Programs Under the Answer Set Semantics for the Semantic Web," *Fundamenta Informaticae*, vol. 82, no. 3, pp. 289 – 310, 2008.

[85] U. Straccia, "Reasoning Within Fuzzy Description Logics," *Journal of Artificial Intelligence Research*, vol. 14, pp. 137 – 166, 2001.

[86] ——, "Towards a Fuzzy Description Logic for the Semantic Web (Preliminary Report)," in *Proceedings of the Second European Semantic Web Conference (ESWC'2005), Heraklion, Crete*, ser. Lecture Notes in Computer Science, A. Gómez-Pérez and J. Euzenat, Eds., vol. 3532. Berlin & Heidelberg, Germany: Springer-Verlag Berlin Heidelberg, 2005, pp. 167 – 181.

[87] T. Lukasiewicz and U. Straccia, "Managing Uncertainty and Vagueness in Description Logics for the Semantic Web," *Journal of Web Semantics*, vol. 6, no. 4, pp. 291 – 308, 2008.

[88] J. Heinsohn, "Probabilistic Description Logics," in *Proceedings of the 10th Annual Conference on Uncertainty in Artificial Intelligence (UAI'94), Seattle, Washington, USA*, R. L. de Mántaras and D. Poole, Eds. San Francisco, California, USA: Morgan Kaufmann Publishers, 1994, pp. 311 – 318.

[89] D. Calvanese, G. De Giacomo *et al.*, "Tractable Reasoning and Efficient Query Answering in Description Logics: The DL-Lite Family," *Journal of Automated Reasoning*, vol. 39, no. 3, pp. 385 – 429, 2007.

[90] F. Baader, D. Calvanese *et al.*, *The Description Logic Handbook: Theory, Implementation and Applications*, 2nd ed. New York, New York, USA: Cambridge University Press, 2010.

[91] A. Borgida, "Description Logics in Data Management," *IEEE Transactions on Knowledge and Data Engineering*, vol. 7, no. 5, pp. 671 – 682, 1995.

[92] H. H. Leitner and M. U. Freeman, "Structured Inheritance Networks and Natural Language Understanding," in *Proceedings of the Sixth International Joint Conference on Artificial Intelligence (IJCAI'79), Tokyo, Japan*, B. G. Buchanan, Ed. San Francisco, California, USA: Morgan Kaufmann Publishers, 1979, pp. 525 – 530.

[93] R. J. Brachman, P. G. Selfridge *et al.*, "Integrated Support for Data Archaeology," in *Proceedings of the 11th National Conference on Artifical Intelligence (AAAI'93), Washington, District of Columbia, USA*, K. Ford, Ed. Menlo Park, California, USA: AAAI Press, 1993, pp. 197 – 211.

[94] M. Grathwohl, F. de Bertrand de Beuvron, and F. Rousselot, "A New Application for Description Logics: Disaster Management," in *Proceedings of the 1999 International Workshop on Description Logics (DL'99), Linköping,*

*Sweden*, ser. CEUR Workshop Proceedings, P. Lambrix, A. Borgida *et al.*, Eds., vol. 22. RWTH Aachen, Aachen, Germany: CEUR-WS.org, 1999. [Online]. Available: http://ceur-ws.org/Vol-22

[95] N. Rychtyckyj, "DLMS: Ten Years of AI for Vehicle Assembly Process Planning," in *Proceedings of the 16th National Conference on Artificial Intelligence (AAAI'99) and 11th Conference on Innovative Applications of Artificial Intelligence (IAAI'99), Orlando, Florida, USA*, J. Hendler and D. Subramanian, Eds. Menlo Park, California, USA: AAAI Press, 1999, pp. 821 – 828.

[96] N. Rychtyckyj, V. Raman *et al.*, "Ontology Re-Engineering: A Case Study from the Automotive Industry," *AI Magazine*, vol. 38, no. 1, pp. 49 – 60, 2017.

[97] J. R. Wright, E. Weixelbaum *et al.*, "A Knowledge-Based Configurator that Supports Sales, Engineering, and Manufacturing at AT&T Network Systems," *AI Magazine*, vol. 14, no. 3, pp. 69 – 80, 1993.

[98] D. L. McGuinness and J. R. Wright, "An Industrial-Strength Description Logic-Based Configurator Platform," *IEEE Intelligent Systems and their Applications*, vol. 13, no. 4, pp. 69 – 77, 1998.

[99] I. Horrocks, O. Kutz, and U. Sattler, "The Even More Irresistible $\mathcal{SROIQ}$," in *Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR'2006), Lake District, United Kingdom*, P. Doherty, J. Mylopoulos, and C. A. Welty, Eds. Menlo Park, California, USA: AAAI Press, 2006, pp. 57 – 67.

[100] B. N. Grosof, I. Horrocks *et al.*, "Description Logic Programs: Combining Logic Programs with Description Logic," in *Proceedings of the 12th International World Wide Web Conference (WWW'2003), Budapest, Hungary*, G. Hencsey, B. White *et al.*, Eds. New York, New York, USA: ACM, 2003, pp. 48 – 57.

[101] F. W. Hartel, S. de Coronado *et al.*, "Modeling a Description Logic Vocabulary for Cancer Research," *Journal of Biomedical Informatics*, vol. 38, no. 2, pp. 114 – 129, 2005.

[102] M. d'Aquin, C. Baldassarre *et al.*, "Characterizing Knowledge on the Semantic Web with Watson," in *Proceedings of the Fifth International Workshop on Evaluation of Ontologies and Ontology-Based Tools (EON'2007), Co-Located with the Sixth International Semantic Web Conference (ISWC'2007), Busan, Korea*, ser. CEUR Workshop Proceedings, R. Garcia-Castro, D. Vrandecic *et al.*, Eds., vol. 329. RWTH Aachen, Aachen, Germany: CEUR-WS.org, 2007, pp. 1 – 10. [Online]. Available: http://ceur-ws.org/Vol-329

[103] F. Baader and W. Nutt, "Basic Description Logics," in *The Description Logic Handbook: Theory, Implementation and Applications*, 2nd ed., F. Baader, D. Calvanese *et al.*, Eds. New York, New York, USA: Cambridge University Press, 2010, pp. 47 – 98.

[104] F. Baader, I. Horrocks *et al.*, "A Basic Description Logic," in *An Introduction to Description Logic*, 1st ed. New York, New York, USA: Cambridge University Press, 2017, ch. 2, pp. 10 – 49.

[105] S. Schlobach, Z. Huang *et al.*, "Debugging Incoherent Terminologies," *Journal of Automated Reasoning*, vol. 39, no. 3, pp. 317 – 349, 2007.

[106] D. Nardi and R. J. Brachman, "An Introduction to Description Logics," in *The Description Logic Handbook: Theory, Implementation and Applications*, 2nd ed., F. Baader, D. Calvanese *et al.*, Eds. New York, New York, USA: Cambridge University Press, 2010, pp. 1 – 44.

[107] F. Baader, R. Küsters, and R. Molitor, "Computing Least Common Subsumers in Description Logics with Existential Restrictions," in *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI'99), Stockholm, Sweden*, T. Dean, Ed. San Francisco, California, USA: Morgan Kaufmann Publishers, 1999, pp. 96 – 103.

[108] F. Baader, "Terminological Cycles in a Description Logic with Existential Restrictions," in *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI'03), Acapulco, Mexico*, G. Gottlob and T. Walsh, Eds. San Francisco, California, USA: Morgan Kaufmann Publishers, 2003, pp. 325 – 330.

[109] F. M. Donini, M. Lenzerini *et al.*, "The Complexity of Concept Languages," *Journal of Information and Computation*, vol. 134, no. 1, pp. 1 – 58, 1997.

[110] Y. Kazakov, "$\mathcal{RIQ}$ and $\mathcal{SROIQ}$ are Harder than $\mathcal{SHOIQ}$," in *Proceedings of the 11th International Conference on Principles of Knowledge Representation and Reasoning (KR'2008), Sydney, Australia*, G. Brewka and J. Lang, Eds. Menlo Park, California, USA: AAAI Press, 2008, pp. 274 – 284.

[111] B. Glimm, I. Horrocks *et al.*, "HermiT: An OWL 2 Reasoner," *Journal of Automated Reasoning*, vol. 53, no. 3, pp. 245 – 269, 2014.

[112] A. Steigmiller, T. Liebig, and B. Glimm, "Konclude: System Description," *Journal of Web Semantics*, vol. 27, no. 1, pp. 78 – 85, 2014.

[113] Y. Kazakov, M. Krötzsch, and F. Simancik, "The Incredible ELK - From Polynomial Procedures to Efficient Reasoning with $\mathcal{EL}$ Ontologies," *Journal of Automated Reasoning*, vol. 53, no. 1, pp. 1 – 61, 2014.

[114] B. Parsia, N. Matentzoglu *et al.*, "The OWL Reasoner Evaluation (ORE) 2015 Competition Report," *Journal of Automated Reasoning*, vol. 59, no. 4, pp. 455 – 482, 2017.

[115] S. O. Hansson, "Reversing the Levi Identity," *Journal of Philosophical Logic*, vol. 22, no. 6, pp. 637 – 669, 1993.

[116] H. Wang, M. Horridge *et al.*, "Debugging OWL-DL Ontologies: A Heuristic Approach," in *Proceedings of the Fourth International Semantic Web Conference (ISWC'2005), Galway, Ireland*, ser. Lecture Notes in Computer Science, Y. Gil, E. Motta *et al.*, Eds., vol. 3729. Berlin & Heidelberg, Germany: Springer-Verlag Berlin Heidelberg, 2005, pp. 745 – 757.

[117] A. Kalyanpur, B. Parsia *et al.*, "Debugging Unsatisfiable Classes in OWL Ontologies," *Journal of Web Semantics*, vol. 3, no. 4, pp. 268 – 293, 2005.

[118] S. Schlobach, "Debugging and Semantic Clarification by Pinpointing," in *Proceedings of the Second European Semantic Web Conference (ESWC'2005), Heraklion, Crete, Greece*, ser. Lecture Notes in Computer Science, A. Gómez-Pérez and J. Euzenat, Eds., vol. 3532. Berlin & Heidelberg, Germany: Springer-Verlag Berlin Heidelberg, 2005, pp. 226 – 240.

[119] P. Plessers and O. D. Troyer, "Resolving Inconsistencies in Evolving Ontologies," in *Proceedings of the Third European Semantic Web Conference (ESWC'2006), Budva, Montenegro*, ser. Lecture Notes in Computer Science, Y. Sure and J. Domingue, Eds., vol. 4011. Berlin & Heidelberg, Germany: Springer-Verlag Berlin Heidelberg, 2006, pp. 200 – 214.

[120] S. Schlobach and R. Cornet, "Non-Standard Reasoning Services for the Debugging of Description Logic Terminologies," in *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI'03), Acapulco, Mexico*, G. Gottlob and T. Walsh, Eds. San Francisco, California, USA: Morgan Kaufmann Publishers, 2003, pp. 355 – 362.

[121] A. Kalyanpur, B. Parsia *et al.*, "Repairing Unsatisfiable Concepts in OWL Ontologies," in *Proceedings of the Third European Semantic Web Conference (ESWC'2006), Budva, Montenegro*, ser. Lecture Notes in Computer Science, Y. Sure and J. Domingue, Eds., vol. 4011. Berlin & Heidelberg, Germany: Springer-Verlag Berlin Heidelberg, 2006, pp. 170 – 184.

[122] Q. Ji, P. Haase *et al.*, "RaDON-Repair and Diagnosis in Ontology Networks," in *Proceedings of the Sixth European Semantic Web Conference on the Semantic Web (ESWC'2009), Heraklion, Crete, Greece*, ser. Lecture Notes in Computer Science, L. Aroyo, P. Traverso *et al.*, Eds., vol. 5554. Berlin & Heidelberg, Germany: Springer-Verlag Berlin Heidelberg, 2009, pp. 863 – 867.

[123] A. A. Kalyanpur, "Debugging and Repair of OWL Ontologies," Ph.D. dissertation, University of Maryland, College Park, 2006.

[124] M. Horridge, B. Parsia, and U. Sattler, "Explaining Inconsistencies in OWL Ontologies," in *Proceedings of the Third International Conference on Scalable Uncertainty Management (SUM'2009), Washington, District of Columbia, USA*, ser. Lecture Notes in Computer Science, L. Godo and A. Pugliese, Eds., vol. 5785.

Berlin & Heidelberg, Germany: Springer-Verlag Berlin Heidelberg, 2009, pp. 124 – 137.

[125] D. A. Plaisted and S. Greenbaum, "A Structure-Preserving Clause Form Translation," *Journal of Symbolic Computation*, vol. 2, no. 3, pp. 293 – 304, 1986.

[126] J. S. C. Lam, D. Sleeman *et al.*, "A Fine-Grained Approach to Resolving Unsatisfiable Ontologies," vol. 10, pp. 62 – 95, 2008.

[127] T. A. Meyer, K. Lee *et al.*, "Finding Maximally Satisfiable Terminologies for the Description Logic $\mathcal{ALC}$," in *First Part of the Proceedings of the 21th National Conference on Artificial Intelligence (AAAI'06), Co-Located with the 18th Innovative Applications of Artificial Intelligence Conference (IAAI'06), Boston, Massachusetts, USA*, A. Cohn, Ed. Menlo Park, California, USA: AAAI Press, 2006, pp. 269 – 274.

[128] F. Baader, I. Horrocks *et al.*, "Reasoning in the $\mathcal{EL}$ Family of Description Logics," in *An Introduction to Description Logic*, 1st ed. New York, New York, USA: Cambridge University Press, 2017, pp. 140 – 167.

[129] S. Colucci, T. D. Noia *et al.*, "Concept Abduction and Contraction in Description Logics," in *Proceedings of the 16th International Workshop on Description Logics (DL'2003), Rome, Italy*, ser. CEUR Workshop Proceedings, D. Calvanese, G. D. Giacomo, and E. Franconi, Eds., vol. 81. RWTH Aachen, Aachen, Germany: CEUR-WS.org, 2003. [Online]. Available: http://ceur-ws.org/Vol-81

[130] G. Teege, "Making the Difference: A Subtraction Operation for Description Logics," in *Proceedings of the Fourth International Conference on Principles of Knowledge Representation and Reasoning (KR'94), Bonn, Germany*, J. Doyle, E. Sandewall, and P. Torasso, Eds. San Francisco, California, USA: Morgan Kaufmann Publishers, 1994, pp. 540 – 550.

[131] F. M. Suchanek, C. Menard *et al.*, "Technical Report: A Language for Combinatorial Creativity," *Telecom ParisTech*, 2016, [Online]. Available: https://suchanek.name, accessed: 2017-12-10.

[132] S. Brandt, R. Küsters, and A.-Y. Turhan, "Approximation and Difference in Description Logics," in *Proceedings of the Eighth International Conference on Principles of Knowledge Representation and Reasoning (KR'2002), Toulouse, France*, D. Fensel, F. Giunchiglia *et al.*, Eds. San Francisco, California, USA: Morgan Kaufmann Publishers, 2002.

[133] P. Gärdenfors and H. Rott, "Belief Revision," in *Epistemic and Temporal Reasoning*, 1st ed., ser. Handbook of Logic in Artificial Intelligence and Logic Programming, D. M. Gabbay, C. J. Hogger, and J. A. Robinson, Eds. Oxford, UK: Oxford University Press, 1995, vol. 4.

[134] A. Fuhrmann, "Theory Contraction Through Base Contraction," *Journal of Philosophical Logic*, vol. 20, no. 2, pp. 175 – 203, 1991.

[135] C. McCamy, "Observation and Measurement of the Appearance of Metallic Materials. Part I. Macro Appearance," *Color Research & Application*, vol. 21, no. 4, pp. 292 – 304, 1996.

[136] F. Sjöqvist and Y. Böttiger, "Historical Perspectives: Drug Interactions – It All Began With Cheese," *Journal of Internal Medicine*, vol. 268, no. 6, pp. 512 – 515, 2010.

[137] M. A. Musen, "The Protégé Project: A Look Back and a Look Forward," *AI Matters*, vol. 1, no. 4, pp. 4 – 12, 2015.

[138] J. Cardoso, "The Semantic Web Vision: Where Are We?" *IEEE Intelligent systems*, vol. 22, no. 5, 2007.

[139] M. Horridge, B. Parsia, and U. Sattler, "Explanation of OWL Entailments in Protégé 4," in *Proceedings of the Poster and Demonstration Session at the Seventh International Semantic Web Conference (ISWC'2008), Karlsruhe, Germany*, ser. CEUR Workshop Proceedings, C. Bizer and A. Joshi, Eds., vol. 401. RWTH Aachen, Aachen, Germany: CEUR-WS.org, 2008. [Online]. Available: http://ceur-ws.org/Vol-401

[140] B. Sertkaya, "OntoComP: A Protégé Plugin for Completing OWL Ontologies," in *Proceedings of the Sixth European Semantic Web Conference (ESWC'2009), Heraklion, Crete, Greece*, ser. Lecture Notes in Computer Science, L. Aroyo, P. Traverso *et al.*, Eds., vol. 5554. Berlin & Heidelberg, Germany: Springer-Verlag Berlin Heidelberg, 2009, pp. 898 – 902.

[141] F. Hanika, G. Wohlgenannt, and M. Sabou, "The uComp Protégé Plugin for Crowdsourcing Ontology Validation," in *Proceedings of the Posters & Demonstrations Track at the 13th International Semantic Web Conference (ISWC'2014), Riva del Garda, Italy*, ser. CEUR Workshop Proceedings, M. Horridge, M. Rospocher, and J. van Ossenbruggen, Eds., vol. 1272. RWTH Aachen, Aachen, Germany: CEUR-WS.org, 2014, pp. 253 – 256. [Online]. Available: http://ceur-ws.org/Vol-1272

[142] M. K. Sarker, A. Krisnadhi *et al.*, "Rule-based OWL Modeling with ROWLTab Protégé Plugin," in *First Part of the Proceedings of the 14th European Semantic Web Conference (ESWC'2017), Portorož, Slovenia*, ser. Lecture Notes in Computer Science, E. Blomqvist, D. Maynard *et al.*, Eds., vol. 10249. Cham, Switzerland: Springer International Publishing, 2017, pp. 419 – 433.

[143] S. Schenk, R. Dividino, and S. Staab, "Using Provenance to Debug Changing Ontologies," *Journal of Web Semantics*, vol. 9, no. 3, pp. 284 – 298, 2011.