NeOn: Lifecycle Support for Networked Ontologies

Integrated Project (IST-2005-027595)

Priority: IST-2004-2.4.7 — "Semantic-based knowledge and content systems"

D1.2.4 Inconsistency-tolerant Reasoning with Networked Ontologies

Deliverable Co-ordinator: **Guilin Qi**

Deliverable Co-ordinating Institution: **University of Karlsruhe**

Other Authors: **Peter Haase (UKarl), Simon Schenk (UKoblenz), Steffen Stadtmüller (UKarl), Pascal Hitzler(UKarl)**

In this deliverable, we discuss the problem of reasoning with inconsistent networked ontologies. We first extend the semantics of description logic $\mathcal{ALC}$ with a four-valued semantics. This will allow us to reasoning with inconsistent ontologies non-trivially. We implement an algorithm for reasoning with the four-valued semantics and provide a prototype. We then propose a bilattice-based semantics to generalize the four-valued semantics. We extend OWL2 to bilattice. The bilattice-based semantics can be used to reasoning with trust information and deal with inconsistency. The bilattice-based semantics is applied to a single ontology which is integrated by networked ontologies. Therefore, we propose another approach for reasoning with distributed ontologies which is based on concept forgetting.

Document Identifier:	NEON/2009/D1.2.4/v1.1	Date due:	February 28, 2009
Class Deliverable:	NEON EU-IST-2005-027595	Submission date:	February 28, 2009
Project start date	March 1, 2006	Version:	v1.1
Project duration:	4 years	State:	Final
		Distribution:	Public

NeOn Consortium

This document is part of the NeOn research project funded by the IST Programme of the Commission of the European Communities by the grant number IST-2005-027595. The following partners are involved in the project:

Open University (OU) – Coordinator Knowledge Media Institute – KMi Berrill Building, Walton Hall Milton Keynes, MK7 6AA United Kingdom Contact person: Martin Dzbor, Enrico Motta E-mail address: {m.dzbor, e.motta}@open.ac.uk	**Universität Karlsruhe – TH (UKARL)** Institut für Angewandte Informatik und Formale Beschreibungsverfahren – AIFB Englerstrasse 11 D-76128 Karlsruhe, Germany Contact person: Peter Haase E-mail address: pha@aifb.uni-karlsruhe.de
Universidad Politécnica de Madrid (UPM) Campus de Montegancedo 28660 Boadilla del Monte Spain Contact person: Asunción Gómez Pérez E-mail address: asun@fi.ump.es	**Software AG (SAG)** Uhlandstrasse 12 64297 Darmstadt Germany Contact person: Walter Waterfeld E-mail address: walter.waterfeld@softwareag.com
Intelligent Software Components S.A. (ISOCO) Calle de Pedro de Valdivia 10 28006 Madrid Spain Contact person: Jesús Contreras E-mail address: jcontreras@isoco.com	**Institut 'Jožef Stefan' (JSI)** Jamova 39 SL–1000 Ljubljana Slovenia Contact person: Marko Grobelnik E-mail address: marko.grobelnik@ijs.si
Institut National de Recherche en Informatique et en Automatique (INRIA) ZIRST – 665 avenue de l'Europe Montbonnot Saint Martin 38334 Saint-Ismier, France Contact person: Jérôme Euzenat E-mail address: jerome.euzenat@inrialpes.fr	**University of Sheffield (USFD)** Dept. of Computer Science Regent Court 211 Portobello street S14DP Sheffield, United Kingdom Contact person: Hamish Cunningham E-mail address: hamish@dcs.shef.ac.uk
Universität Kolenz-Landau (UKO-LD) Universitätsstrasse 1 56070 Koblenz Germany Contact person: Steffen Staab E-mail address: staab@uni-koblenz.de	**Consiglio Nazionale delle Ricerche (CNR)** Institute of cognitive sciences and technologies Via S. Marino della Battaglia 44 – 00185 Roma-Lazio Italy Contact person: Aldo Gangemi E-mail address: aldo.gangemi@istc.cnr.it
Ontoprise GmbH. (ONTO) Amalienbadstr. 36 (Raumfabrik 29) 76227 Karlsruhe Germany Contact person: Jürgen Angele E-mail address: angele@ontoprise.de	**Food and Agriculture Organization of the United Nations (FAO)** Viale delle Terme di Caracalla 00100 Rome Italy Contact person: Marta Iglesias E-mail address: marta.iglesias@fao.org
Atos Origin S.A. (ATOS) Calle de Albarracín, 25 28037 Madrid Spain Contact person: Tomás Pariente Lobo E-mail address: tomas.parientelobo@atosorigin.com	**Laboratorios KIN, S.A. (KIN)** C/Ciudad de Granada, 123 08018 Barcelona Spain Contact person: Antonio López E-mail address: alopez@kin.es

Work package participants

The following partners have taken an active part in the work leading to the elaboration of this document, even if they might not have directly contributed to the writing of this document or its parts:

- UKarl
- UKoblenz

Change Log

Version	Date	Amended by	Changes
0.1	10-11-2008	Guilin Qi	Create the deliverable
0.2	03-12-2008	Simon Schenk	Add chapter on paraconsistent and bilattice based reasoning
0.3	27-01-2009	Guilin Qi	Add chapter on forgetting-based reasoning
0.4	06-02-2009	Steffen Stadtmüller	Add chapter on implementation
0.5	07-02-2009	Guilin Qi	Add abstract
0.6	18-02-2008	Simon Schenk	Revise chapter on paraconsistent and bilattice based reasoning
0.6	07-02-2009	Guilin Qi	Add Conclusion
0.7	01-03-2009	Guilin Qi	Revise the deliverable based on Peter's comments
0.8	01-03-2009	Guilin Qi and Simon Schenk	Finalize the deliverable

Executive Summary

Next generation semantic applications are characterized by a large number of ontologies, some of them constantly evolving. As the complexity of semantic applications increases, more and more knowledge are embedded in applications, typically drawn from a wide variety of sources. This new generation of applications thus likely rely on ontologies embedded in a network of already existing ontologies. Ontologies and metadata have to be kept up to date when application environments and users' needs change. One of the major challenges in managing these networked and dynamic ontologies is to handle potential inconsistencies.

We proposed a general approach to repairing a single ontology in NeOn deliverable D1.2.1 [QHJ07] and provided evaluation results on our approach in NeOn deliverable D1.2.2 [QHJV07]. We then gave an approach to repairing networked ontologies in NeOn deliverable D1.2.3 [QHJ08]. However, it is not always desirable to drop information in ontologies to resolve inconsistencies. Instead, an inconsistency-tolerant approach may be interesting in some cases. For example, when integrating ontologies connected by mappings, we may not have right to repair either ontologies or mappings. It has also been argued that repairing a knowledge base may cause unintended loss of important information by some researchers [Hun06].

In this deliverable, we discuss the problem of reasoning with inconsistent networked ontologies. We first extend the semantics of description logic $\mathcal{ALC}$ with a four-valued semantics. It introduces additional truth values standing for *unkown* (i.e. neither true nor false) and *contradiction* (i.e. both true and false). This will allow us to reasoning with inconsistent ontologies non-trivially. Syntactically, four-valued logic is very similar to classical logic. Syntactically, $\mathcal{ALC}_4$ hardly differs from $\mathcal{ALC}$. Complex concepts and assertions are defined in exactly the same way. For class inclusion, however, the question arises how to interpret the underlying implication connective in the four-valued setting. We thus allow three kinds of class inclusions: $C \mapsto D, C \supset D, C \rightarrow D$, called material inclusion axiom, internal inclusion axiom, and strong inclusion axiom, respectively. We implement an algorithm for reasoning with the four-valued semantics and provide a prototype. We then propose a bilattice-based semantics to generalize the four-valued semantics. The bilattice-based semantics is very useful when we want to integrate networked ontologies. We propose an approach for obtaining bilattices. We extend $\mathcal{SROIQ}$, the description logic underlying the proposed OWL2 [GM08], to $\mathcal{SROIQ} - \mathcal{T}$ evaluated on a logical bilattice. The bilattice-based semantics can be used to reasoning with trust information and deal with inconsistency. The bilattice-based semantics is applied to a single ontology which is integrated by networked ontologies. Therefore, we propose another approach for reasoning with distributed ontologies which is based on concept forgetting.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 The NeOn Big Picture

Next generation semantic applications will be characterized by a large number of ontologies, some of them constantly evolving. As the complexity of semantic applications increases, more and more knowledge will be embedded in applications, typically drawn from a wide variety of sources. This new generation of applications will thus likely rely on ontologies embedded in a network of already existing ontologies. Ontologies and metadata will have to be kept up to date when application environments and users' needs change. We argue that in this scenario it will become prohibitively expensive for people to directly adopt the current approach to semantic integration, where the expectation is to produce a single, globally consistent semantic model that serves the needs of application developers and fully integrates a number of pre-existing ontologies. In contrast to the current model, future applications will very likely rely on networks of contextualized ontologies, which are usually locally, but not globally consistent.

This report is part of the work performed in WP 1 on "Dynamics of Networked Ontologies". The goal of this work package is to develop an integrated approach for the evolution process of networked ontologies and related metadata. As shown in Figure 1.1, WP1 belongs to the central part of the research and development WPs in NeOn. The tasks of WP1 are heavily inter-related with other work packages. For the individual phases of the process we will develop new methods that consider the complex relationships in a network of ontologies. These include dependencies, mappings, different versions and also take possible inconsistencies into account.

Specific goals in this workpackage include support for:

1. representing, managing and interpreting dependencies between multiple networked ontologies
2. evolution of networked ontologies in exploiting various models of change propagation, which have different applicabilities depending on the model of coordination and control
3. maintaining partial/local consistency of a set of networked ontologies, which might not be globally consistent
4. evolving metadata along with changing ontologies and predicting future structural changes in ontologies.

1.2 Motivation and Goals of this Deliverable

Real knowledge bases and data for Semantic Web applications will rarely be perfect. They will be distributed and multi-authored. They will be assembled from different sources and reused. It is unreasonable to expect such reaslistic knowledge bases to be always logically consistent, and it is therefore important to study

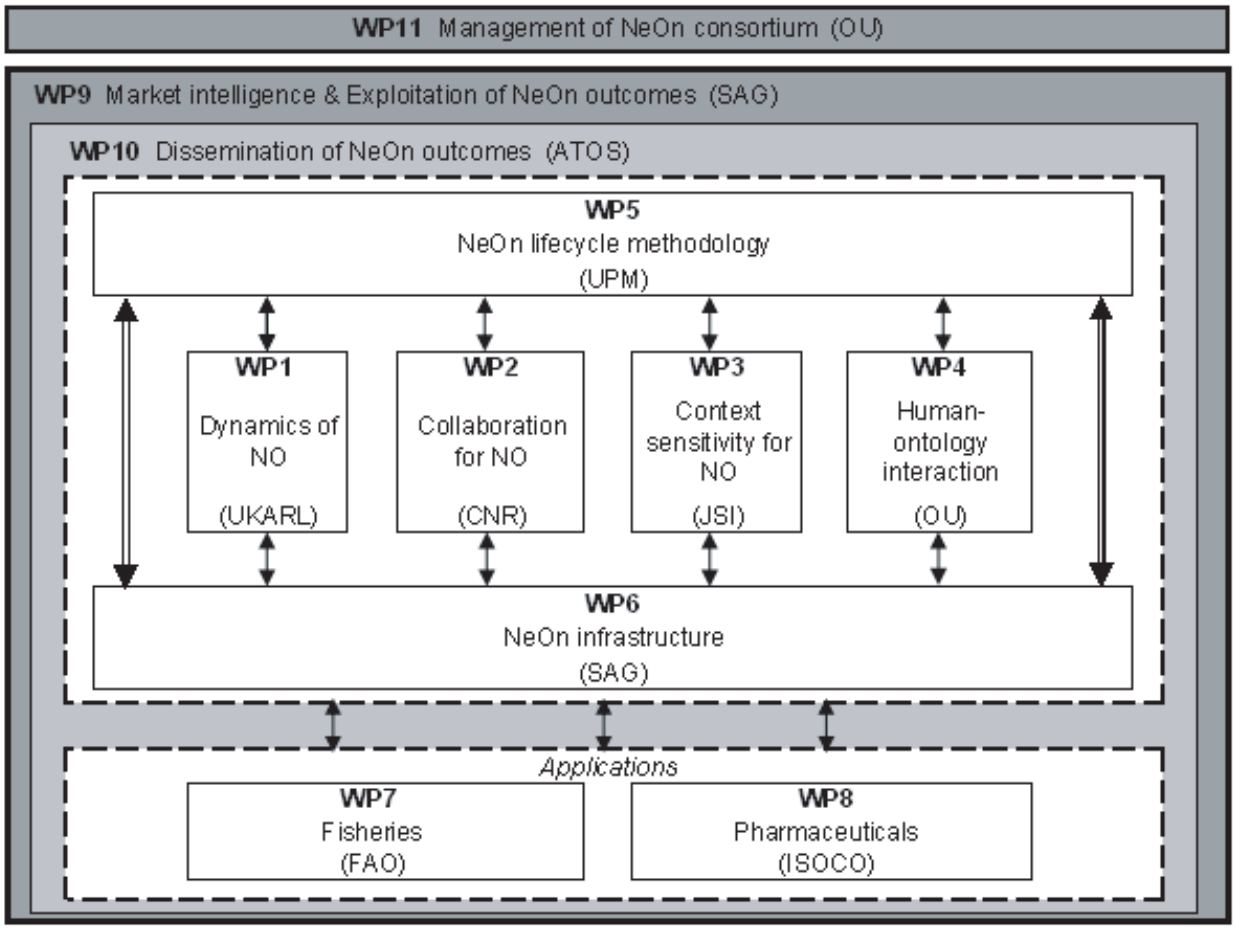


Figure 1.1: Relationships between different workpackages in NeOn

ways of dealing with inconsistent knowledge. This is particularly important if the full power of logic-based approaches like the Web Ontology Language OWL shall be employed, as classical logic breaks down in the presence of inconsistent knowledge. The study of inconsistency handling in Artificial Intelligence has a long tradition, and corresponding results are recently being transferred to description logics, which underlay OWL. Two fundamentally different approaches can be distinguished. The first is based on the assumption that inconsistencies indicate erroneous data which is to be repaired in order to obtain a consistent knowledge base, e.g. by selecting consistent subsets for the reasoning process [SC03, HvHH$^+$05]. The other approach yields to the insight that inconsistencies are a natural phenomenon in realistic data which are to be handled by a logic which tolerates it [PS89, Str97, HvHtT05, MLL06]. Such logics are called paraconsistent, and the most prominent of them are based on the use of additional truth values standing for *underdefined* (i.e. neither true nor false) and *overdefined* (or *contradictory*, i.e. both true and false). Such logics are appropriately called *four-valued logics* [Bel77]. We believe that either of the approaches is useful, depending on the application scenario.

We proposed a general approach to repairing a single ontology in NeOn deliverable D1.2.1 [QHJ07] and provided evaluation results on our approach in NeOn deliverable D1.2.2 [QHJV07]. We then gave an approach to repairing networked ontologies in NeOn deliverable D1.2.3 [QHJ08]. However, it is not always desirable to drop information in ontologies to resolve inconsistencies. Instead, an inconsistency-tolerant approach may be interesting in some cases. For example, when integrating ontologies connected by mappings, we may not have right to repair either ontologies or mappings. It has also been argued that repairing a knowledge base may cause unintended loss of important information by some researchers [Hun06].

In this deliverable, we discuss the problem of reasoning with inconsistent networked ontologies. We first extend the semantics of description logic $\mathcal{ALC}$ with a four-valued semantics. It introduces additional truth values standing for *unknown* (i.e. neither true nor false) and *contradiction* (i.e. both true and false). This will allow us to reasoning with inconsistent ontologies non-trivially. Syntactically, $\mathcal{ALC}_4$ hardly differs from $\mathcal{ALC}$. Complex concepts and assertions are defined in exactly the same way. For class inclusion, however, the question arises how to interpret the underlying implication connective in the four-valued setting. We thus allow three kinds of class inclusions: $C \mapsto D, C \supset D, C \rightarrow D$, called material inclusion axiom, internal inclusion axiom, and strong inclusion axiom, respectively. We implement an algorithm for reasoning with the four-valued semantics and provide a prototype. We then propose a bilattice-based semantics to generalize the four-valued semantics. The bilattice-based semantics is very useful when we want to integrate networked ontologies. We propose an approach for obtaining bilattices. We extend $\mathcal{SROIQ}$, the description logic underlying the proposed OWL2 [GM08], to $\mathcal{SROIQ} - \mathcal{T}$ evaluated on a logical bilattice. The bilattice-based semantics can be used to reasoning with trust information and deal with inconsistency. However, it is only applied to a single ontology which is integrated by networked ontologies. Therefore, we propose an approach for reasoning with distributed ontologies which is based on concept forgetting.

1.3 Overview of the Deliverable

This deliverable is structured as follows. We first present four-valued description logic $\mathcal{ALC}$ and provide a description of an implementation of a paraconsistent reasoning algorithm and a plugin to the NeOn toolkit in Chapter 2. We then generalize the four-valued semantics to bilattice based reasoning with description logics in Chapter 3, and discuss applications of the logical bilattice. Finally, we propose another approach for reasoning with distributed ontologies which is based on concept forgetting in Chapter 4.

Chapter 2

Paraconsistent Reasoning with Description Logics

Real knowledge bases and data for Semantic Web applications will rarely be perfect. They will be distributed and multi-authored. They will be assembled from different sources and reused. It is unreasonable to expect such realistic knowledge bases to be always logically consistent, and it is therefore important to study ways of dealing with inconsistent knowledge. This is particularly important if the full power of logic-based approaches like the Web Ontology Language OWL [PSH04] shall be employed, as classical logic breaks down in the presence of inconsistent knowledge.

The study of inconsistency handling in Artificial Intelligence has a long tradition, and corresponding results are recently being transferred to description logics, which underlay OWL. Two fundamentally different approaches can be distinguished. The first is based on the assumption that inconsistencies indicate erroneous data which is to be repaired in order to obtain a consistent knowledge base, e.g. by selecting consistent subsets for the reasoning process [SC03, HvHH$^+$05]. The other approach yields to the insight that inconsistencies are a natural phenomenon in realistic data which are to be handled by a logic which tolerates it [PS89, Str97, MLL06]. Such logics are called paraconsistent, and the most prominent of them are based on the use of additional truth values standing for undefined (i.e. neither true nor false) and overdefined (or contradictory, i.e. both true and false). Such logics are appropriately called four-valued logics [Bel77]. We believe that either of the approaches is useful, depending on the application scenario.

In this chapter, we contribute to the paraconsistency approach. We indeed extend on the preliminary work in [MLL06], which has the following features.

- It is grounded in prominent research results from Artificial Intelligence [AA98].
- It is very flexible in terms of design choices which can be made when developing a paraconsistent description logic. This concerns the issues arising from the fact that there are different ways of defining the notion of logical implication in four-valued logics. The approach which we follow allows the full and simultaneous use of the different notions of implication.
- It does not increase worst-case computational complexity of reasoning if compared to standard reasoning methods for consistent knowledge bases.

The chapter is structured as follows. We first review briefly preliminaries in Section 2.1. In Section 2.2 we then describe the syntax and semantics of the paraconsistent description logic which we will use. We provide an algorithm for computing the four-valued semantics in Section 2.3. Finally, we give an implementation of the algorithm in Section 2.5.

2.1 Preliminaries

2.1.1 The Description Logic $\mathcal{ALC}$

We briefly review notation and terminology of the description logic $\mathcal{ALC}$, but we basically assume that the reader is familiar with description logics. For comprehensive background reading, please refer to [BCM+03].

Table 2.1: $\mathcal{ALC}$

Constructor Name	Syntax	Semantics
atomic concept	A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
abstract role	R_A	$R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
individuals	I	$o^{\mathcal{I}} \in \Delta^{\mathcal{I}}$
top concept	$\top$	$\Delta^{\mathcal{I}}$
bottom concept	$\bot$	$\varnothing$
conjunction	$C \sqcap C$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
disjunction	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
negation	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
exists restriction	$\exists R.C$	$\{x \mid \exists y, (x,y) \in R^{\mathcal{I}} \text{and} y \in C^{\mathcal{I}}\}$
value restriction	$\forall R.C$	$\{x \mid \forall y, (x,y) \in R^{\mathcal{I}} \text{implies} y \in C^{\mathcal{I}}\}$

Axiom Name	Syntax	Semantics
concept inclusion	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq d^{\mathcal{I}}$
concept assertion	$C(a)$	$a^{\mathcal{I}} \in C^{\mathcal{I}}$
role assertion	$R(a,b)$	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$

We assume that we are given a set of atomic concepts (or concept names), a set of roles (or role names), and a set of individuals. With the symbols and $\bot$ we furthermore denote the top concept and the bottom concept, respectively.
Complex concepts in $\mathcal{ALC}$ can be formed from these inductively as follows.

1. $\top$, $\bot$, and each atomic concept are concepts;
2. If C, D are concepts, then $C \sqcap D$, $C \sqcup D$, and $\neg C$ are concepts;
3. If C is a concept and R is a role, then $\forall R.C$ and $\exists R.C$ are concepts.

An $\mathcal{ALC}$ ontology consists of a set of assertions, called the ABox of the ontology, and a set of inclusion axioms, called the T Box of the ontology. Assertions are of the form $C(a)$ or $R(a,b)$, where a, b are individuals and C and R are concepts and roles, respectively. Inclusion axioms are of the form $C \sqsubseteq D$, where C and D are concepts. Informally, an assertion $C(a)$ means that the individual a is an instance of concept C, and an assertion $R(a,b)$ means that individual a is related with individual b via the property R. The inclusion axiom $C \sqsubseteq D$ means that each individual of C is an individual of D.
The formal definition of the (model-theoretic) semantics of $\mathcal{ALC}$ is given by means of interpretations $I = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consisting of a non-empty domain $\Delta^{\mathcal{I}}$ and a mapping $\cdot^{\mathcal{I}}$ satisfying the conditions in Table 2.1.1, interpreting concepts as subsets of the domain and roles as binary relations on the domain. An interpretation satisfies an $\mathcal{ALC}$ ontology (i.e. is a model of the ontology) iff it satisfies each axiom in both the ABox and the T Box. An ontology is called satisfiable (unsatisfiable) iff there exists (does not exist) such a model. In $\mathcal{ALC}$, reasoning tasks, i.e. the derivation of logical consequences, can be reduced to satisfiability checking of ontologies [BCM+03, HPS04].

Table 2.2: Truth table for 4-valued connectives

α	f	f	f	f	t	t	t	t	$\top$	$\top$	$\top$	$\top$	$\bot$	$\bot$	$\bot$	$\bot$
β	f	t	$\top$	$\top$	f	t	$\top$	$\top$	f	t	$\top$	$\top$	f	t	$\top$	$\top$
$\neg\alpha$	t	t	t	t	f	f	f	f	$\top$	$\top$	$\top$	$\top$	$\bot$	$\bot$	$\bot$	$\bot$
$\alpha \wedge \beta$	f	f	f	f	f	t	$\top$	$\bot$	f	$\top$	$\top$	f	f	$\bot$	$\bot$	f
$\alpha \vee \beta$	f	t	$\top$	$\top$	t	t	t	t	$\top$	t	$\top$	t	$\bot$	t	t	$\bot$
$\alpha \mapsto \beta$	t	t	t	t	f	t	$\top$	$\top$	$\top$	t	$\top$	t	$\bot$	t	t	$\bot$
$\alpha \supset \beta$	t	t	t	t	f	t	$\top$	$\top$	f	t	$\top$	$\bot$	t	t	t	t
$alpha \rightarrow \beta$	t	t	t	t	f	t	f	$\top$	f	t	$\top$	$\bot$	$\bot$	t	$\bot$	t

2.1.2 Four-valued Logic

The major studies of four-valued logics have been carried out in the setting of propositional logic. We will very briefly review the preliminaries which set the state for the four-valued version of $\mathcal{ALC}$ which we will present later.

The idea of four-valued logic is based on the idea of having four truth values, instead of the classical two. The four truth values stand for true, false, unknown (or undefined) and both (or overdefined, contradictory). We use the symbols t, f, $\bot$, $\top$, respectively, for these truth values, and the set of these four truth values is denoted by $\mathcal{FOUR}$. The truth value $\top$ stands for contradictory information, hence four-valued logic lends itself to dealing with inconsistent knowledge. The value $\top$ thus can be understood to stand for true and false, while $\bot$ stands for neither true nor false, i.e. for the absence of any information about truth or falsity.

Syntactically, four-valued logic is very similar to classical logic. Care has to be taken, however, in defining meaningful notions of implication, as there are several ways to do this. Indeed, there are three major notions of implication in the literature, all of which we will employ in our approach. The logical connectives we allow are thus negation $\neg$, disjunction $\vee$, conjunction $\wedge$, material implication $\mapsto$, internal implication $\supset$, and strong implication $\rightarrow$. We will discuss them in detail later on as the presence of all three implications is crucial for our approach.

Four-valued interpretations for formulae (i.e. 4-interpretations) are obviously mappings from formulae to (the set of four) truth values, respecting the truth tables for the logical connectives, as detailed in Table 2.1.1. Four-valued models (4-models) are defined in the obvious way, as follows, where t and $\top$ are the designated truth values.

Definition 1 *Let $\mathcal{I}$ be a 4-interpretation, let Σ be a theory (i.e. set of formulae) and let ϕ be a formula in four-valued logic. Then $\mathcal{I}$ is a 4-model of ϕ if and only if $\mathcal{I}(\phi) \in \{t, \top\}$. $\mathcal{I}$ is a 4-model of Σ if and only if $\mathcal{I}$ is a 4-model of each formula in Σ. Σ four-valued entails ϕ, written $\Sigma \models_4 \phi$, if and only if every 4-model of Σ is a 4-model of ϕ.*

Proposition 1 *We note the following general properties.*

- *The language $L = \{\neg, \vee, \wedge, \supset, \bot, \top\}$ is functional complete for the set $\mathcal{FOUR}$ of truth values, i.e. every function from $\mathcal{FOUR}^n$ to $\mathcal{FOUR}$ is representable by some formula in L [AA98, Theorem 12].*
- *Any formula containing only connectives from $\{\neg, \vee, \wedge, \supset\}$ always has a four- valued model.*

Some general remarks about the different notions of implication are in order. They are the major notions of implication used in the literature, and are discussed in detail in [AA98, AA96]. The basic rationales behind them are the following: Material implication can be defined by means of negation and disjunction as known from classical logic. However, it does not satisfy Modus Ponens or the deduction theorem, and is thus of limited use as an implication in the intuitive sense. Internal implication satisfies Modus Ponens and the deduction theorem, but cannot be defined by means of other connectives. Furthermore, internal implication does not satisfy contraposition. Strong implication is stronger than internal implication, in that it additionally

Table 2.3: Semantics of $\mathcal{ALC}_4$ Concepts

Constructor Syntax	Semantics
A	$A = \langle P, N\rangle \vert P, N \subseteq \Delta^{\mathcal{I}}$
R	$R^{\mathcal{I}} = \langle R_P, R_N\rangle \vert R_P, R_N \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
o	$o^{\mathcal{I}} \in \Delta^{\mathcal{I}}$
$\top$	$\langle \Delta^{\mathcal{I}}, \varnothing\rangle$
$\bot$	$\langle \varnothing, \Delta^{\mathcal{I}}\rangle$
$C_1 \sqcap C_2$	$\langle P_1 \cap P_2, N_1 \cup N_2\rangle$, if $C_i = \langle P_i, N_i\rangle$ for $i = 1, 2$
$C_1 \sqcup C_2$	$\langle P_1 \cup P_2, N_1 \cap N_2\rangle$, if $C_i = \langle P_i, N_i\rangle$ for $i = 1, 2$
$\neg C$	$(\neg C)^{\mathcal{I}} = \langle N, P\rangle$, if $C^{\mathcal{I}} = \langle P, N\rangle$
$\exists R.C$	$\langle \{x \vert \exists y : (x, y) \in proj^+(R^{\mathcal{I}}) \wedge y \in proj^+(c^{\mathcal{I}})\}$, $\{x \vert \forall y : (x, y) \in proj^+(R^{\mathcal{I}}) \wedge y \in proj^-(c^{\mathcal{I}})\}\rangle$
$\forall R.C$	$\langle \{x \vert \forall y : (x, y) \in proj^+(R^{\mathcal{I}}) \wedge y \in proj^+(c^{\mathcal{I}})\}$, $\{x \vert \exists y : (x, y) \in proj^+(R^{\mathcal{I}}) \wedge y \in proj^-(c^{\mathcal{I}})\}\rangle$

satisfies contraposition. Indeed, an alternative view on the truth tables for the implication connectives is as follows.

$\phi \mapsto \psi$	is definable as	$\neg\phi \vee \psi$.	(Material Implication)
$\phi \supset \psi$	evaluates to	$\begin{cases} \psi \text{ if } \phi \in \{t, \top\} \\ t \text{ if } \phi \in \{f, \bot\} \end{cases}$	(Internal Implication)
$\phi \rightarrow \psi$	is defineable as	$(\phi \supset \psi) \wedge (\neg\psi \supset \neg\phi)$	(Strong Implication)

Further properties of the implication connectives are summarized in the following proposition (as shown in [AA98, Corollary 9] and [AA96]).

Proposition 2 *The following claims hold, where Γ is a theory and ϕ, ψ are formulae.*

- *Internal implication is not definable in terms of the connectives $\neg, \vee, \wedge$.*
- $\Gamma, \phi \models_4 \psi$ *iff* $\Gamma \models_4 \phi \supset \psi$.
- *If* $\Gamma \models_4 \psi$ *and* $\gamma \models_4 \psi \supset \phi$ *then* $\Gamma \models_4 \phi$.
- $\psi \rightarrow \psi$ *implies that* $\neg\phi \rightarrow \neg\psi$.

Apart from the formal properties of the different notions of implication, it is obviously important to consider their intuitive meaning and their usefulness for knowledge base modelling. We will discuss this in detail in the next section.

2.2 The Four-valued Description Logic $\mathcal{ALC}_4$

We describe the syntax and semantics of our four-valued description logic $\mathcal{ALC}_4$. The approach is fairly standard apart from the fact that we allow the simultaneous use of all three notions of implication. We will thus devote significant space to a detailed discussion of the intuitions behind these different implications.

Syntactically, $\mathcal{ALC}_4$ hardly differs from $\mathcal{ALC}$. Complex concepts and assertions are defined in exactly the same way. For class inclusion, however, the question arises how to interpret the underlying implication connective in the four-valued setting. We thus allow three kinds of class inclusions, corresponding to the three implication connectives we have discussed. They are as follows, $C \mapsto D, C \supset D, C \rightarrow D$, called material inclusion axiom, internal inclusion axiom, and strong inclusion axiom, respectively.

Table 2.4: Semantics of inclusion axioms in $\mathcal{ALC}_4$

Axiom Name	Syntax	Semantics
material inclusion	$C_1 \mapsto C_2$	$\Delta^{\mathcal{I}} \setminus proj^-(C_1^{\mathcal{I}}) \subseteq proj^+(C_2^{\mathcal{I}})$
internal inclusion	$C_1 \supset C_2$	$proj^+(C_1^{\mathcal{I}}) \subseteq proj^+(C_2^{\mathcal{I}})$
strong inclusion	$C_1 \rightarrow C_2$	$proj^+(C_1^{\mathcal{I}}) \subseteq proj^+(C_2^{\mathcal{I}}) \wedge proj^-(C_2^{\mathcal{I}}) \subseteq proj^-(C_1^{\mathcal{I}})$
internal inclusion	$C(a)$	$a^{\mathcal{I}} \in proj^+(C^{\mathcal{I}})$
role assertion	$R(a,b)$	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \in proj^+(R^{\mathcal{I}})$

Semantically, interpretations map individuals to elements of the domain of the interpretation, as usual. For concepts, however, we need to make modifications to the notion of interpretation in order to allow for reasoning with inconsistencies.

Intuitively, in four-valued logic we need to consider four situations which can occur in terms of containment of an individual in a concept: (1) we know it is contained, (2) we know it is not contained, (3) we have no knowledge whether or not the individual is contained, (4) we have contradictory information, namely that the individual is both contained in the concept and not contained in the concept. There are several equivalent ways how this intuition can be formalized, one of which is described in the following.

For a given domain $\Delta^{\mathcal{I}}$ and a concept C, an interpretation over $\Delta^{\mathcal{I}}$ assigns to C a pair $\langle P, N\rangle$ of (not necessarily disjoint) subsets of $\Delta^{\mathcal{I}}$. Intuitively, P is the set of elements known to belong to the extension of C, while N is the set of elements known to be not contained in the extension of C. For simplicity of notation, we define functions $proj^+(\cdot)$ and $proj^-(\cdot)$ by $proj^+\langle P, N\rangle = P$ and $proj^-\langle P, N\rangle = N$.

Formally, a four-valued interpretation is a pair $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ with $\Delta^{\mathcal{I}}$ as domain, where $\cdot^{\mathcal{I}}$ is a function assigning elements of $\Delta^{\mathcal{I}}$ to individuals, and subsets of $(\Delta^{\mathcal{I}})^2$ to concepts, such that the conditions in Table 2.2 are satisfied. Note that the conditions in Table 2.2 for role restrictions are designed in such a way that the logical equivalences $\neg(\forall R.C) = \exists R.(\neg C)$ and $\neg(\exists R.C) = \forall R.(\neg C)$ are retained - this is the most convenient way for us for handling role restrictions, as it will allows for a straightforward translation from $\mathcal{ALC}_4$ to classical $\mathcal{ALC}$. Note also that for roles we actually require only the positive part of the extension - we nevertheless require interpretations to assign pairs of sets to roles, which is a technical formality to retain consistency of notation with possible extensions to more expressive description logics (see [MLL06]).

Obviously, under the constraints $P \cap N = \varnothing$ and $P \cup N = \Delta$, four-valued interpretations become just standard two-valued interpretations.

The correspondence between truth values from $\mathcal{FOUR}$ and concept extensions is the obvious one: For instances $a \in \Delta^{\mathcal{I}}$ and concept name C we have

- $C^{\mathcal{I}}(a) = t(\top)$, iff $a^{\mathcal{I}} \in proj^+(C^{\mathcal{I}})$ and $a^{\mathcal{I}} \notin (\in) proj^-(C^{\mathcal{I}})$,
- $C^{\mathcal{I}}(a) = f(\bot)$, iff $a^{\mathcal{I}} \in proj^+(C^{\mathcal{I}})$ and $a^{\mathcal{I}} \in (\notin) proj^-(C^{\mathcal{I}})$,

When defining the semantics as we just did, we ensure that a number of useful equivalences from classical logic hold, as follows.

Proposition 3 *For any four-valued interpretation $\mathcal{I}$ and concepts C, D, the following claims hold.*
$(C \sqcap \top)^{\mathcal{I}} = C^{\mathcal{I}}$,
$(C \sqcup \top)^{\mathcal{I}} = \top^{\mathcal{I}}$,
$(C \sqcap \bot)^{\mathcal{I}} = \bot^{\mathcal{I}}$,
$(C \sqcup \bot)^{\mathcal{I}} = C^{\mathcal{I}})$,
$(\neg\neg C)^{\mathcal{I}} = C^{\mathcal{I}}$,
$(\neg\top)^{\mathcal{I}} = \bot^{\mathcal{I}}$,
$(\neg\bot)^{\mathcal{I}} = \top^{\mathcal{I}}$,
$(\neg(C \sqcup D))^{\mathcal{I}} = (\neg C \sqcap \neg D)^{\mathcal{I}}$,

$(\neg(C \sqcap D))^{\mathcal{I}} = (\neg C \sqcup \neg D)^{\mathcal{I}}$,
$(\neg(\forall R.C))^{\mathcal{I}} = (\exists R.\neg C)^{\mathcal{I}}$,
$(\neg(\exists R.C))^{\mathcal{I}} = (\forall R.\neg C)^{\mathcal{I}}$.

We now come to the semantics of the three different types of inclusion axioms. It is formally defined in Table 2.2 (together with the semantics of concept assertions). We say that a four-valued interpretation I satisfies a four-valued ontology O (i.e. is a model of it) iff it satisfies each assertion and each inclusion axiom in O. An ontology O is satisfiable (unsatisfiable) iff there exists (does not exist) such a model.

With the formal definitions out of the way, it remains to address the intuitions underlying the different inclusion axioms. These intuitions are evidenced by the formal properties of the underlying implications as discussed above as well as the behavior of the implications in practice. We actually foresee a possible workflow for handling inconsistent ontologies, as follows. In a first step, inclusion axioms are classified into the three types of four-valued inclusion axioms available. Then four-valued reasoning is performed based on the classification, in order to arrive at a meaningful 4-valued conclusion. The question, how such a classification can be performed, will not be addressed in this chapter. It constitutes a separate substantial piece of work which is under investigation by the authors. A combination of automated detection and a userinteraction process may be the most workable solution, where the user-interaction process may be guided by the intuitive explanations which we will now give for the three types of inclusion.

Strong inclusion respects the deduction theorem and contraposition reasoning. In a paraconsistent context, it is thus the inclusion to be used for universal truth, such as Square $\mapsto$ FourEdged.

Internal inclusion propagates contradictory information forward, but not backward as it does not allow for contraposition reasoning. It can thus be characterized as a brave way of handling inconsistency. It should be used whenever it is important to infer the consequent even if the antecedent may be contradictory. To give an example, consider a robot fault diagnosis system and an axiom stating that oil leakage is indicative of a robot malfunction. Obviously, it is important to check on a possible malfunction even in case there is contradictory information about an oil leakage. In a paraconsistent context, the axiom is thus best modeled by means of internal inclusion, i.e. as OilLeakage $\supset$ RobotMalfunction.

Material inclusion is cautious in the sense that contradictory information is not propagated. The intuition behind material inclusion becomes apparent by studying the truth table for material implication: $a \rightarrow b$ indicates that the only way for b to be not true (i.e. to be f or $\bot$) is if there is information of falsity of a (i.e. it is f or $\top$). This kind of modeling becomes important if an inclusion has to be second-guessed e.g. after a merging of knowledge bases. Consider, for example, an ontology about marathon runs containing the axiom Healthy $\rightarrow$ MarathonParticipant which is supposed to say that somebody (i.e. a person who has signed up for a run) participates in a marathon if he checks out to be healthy. The axiom is reasonable if the domain is for the management of marathon participants' data only. Now imagine that this ontology is merged with other sports knowledge bases, e.g. a boxing domain. It is wrong to infer that every healthy boxer will participate in the marathon, so the original axiom will likely lead to contradictions. We propose to handle this kind of information by modelling the axiom as material inclusion, i.e. as Healthy $\mapsto$ MarathonParticipant, which will indeed not infer participation from a positive health status. However, the weak form of contraposition reasoning featured by material inclusion results in the following situation: If an individual is not known to be contained in MarathonParticipant, then it is known to be not Healthy, resulting in a possible contradiction on health status while avoiding contradiction in terms of marathon participation, which may be preferred in the domain. Material inclusion may thus propagate contradictory information backwards (to the antecedent), while internal inclusion may propagate contradictory information forward (to the consequent).

We remark here that different inclusion axioms provide ontology engineers with a flexible way to define different ontologies according to the intuition explained above. In case only one kind of inclusion shall be used, we recommend to use strong inclusion, as it should serve the ontology engineer's original intention most closely. To give an example, consider the inconsistent subontology of BuggyPolicy2 (with additional assertions) which says "GeneralReliabilityUsernamePolicy (G for short) is a subset of Reliable, G and Messaging are disjoint, Reliable is a subset of Messaging, p_1 is an individual of G and p_2 is an individual of Reliable".

Using strong inclusion results in the ontology $\{R \mapsto M, G \mapsto R, M \mapsto \neg G, G \mapsto \neg M, G(p_1), R(p_2)\}$, where we use obvious abbreviations for the class names. Under the semantics of strong inclusion, $M(p_1), R(p_1), M(p_2), \neg G(p_1)$, and $\neg M(p_1)$ hold, but $G(p_2)$ does not hold. This example shows that our four-valued semantics can give meaningful answers when an ontology is inconsistent, while classical semantics fails to do so.

The truth values $t, f, \top, \bot$ form the smallest, non-trivial logical bilattice $\mathcal{FOUR}$. Logical bilattices are algebraic structures, which can be used to formalize many logical formalisms. Generally, they are not limited in size. Hence, we can generalize the results from this sections to multi valued logics. In this section we have formalized class membership using membership in two sets. In the general case, such a fixed number if not known. Hence, in the following we will model classes and properties as functions mapping from (pairs of) individuals to truth values taken from a logical bilattice.

2.3 An Algorithm to Compute the Four-valued Semantics

It is a pleasing property of $\mathcal{ALC}4$, that it can be translated easily into classical $\mathcal{ALC}$, such that paraconsistent reasoning can be simulated by using standard $\mathcal{ALC}$ reasoning algorithms.

Definition 2 *(Concept transformation) For any given concept C, its transformation $\pi(C)$ is the concept obtained from C by the following inductively defined transformation.*

- *If $C = A$ for A an atomic concept, then $\pi(C) = A^+$, where A^+ is a new concept;*
- *If $C = \neg A$ for A an atomic concept, then $\pi(C) = A^-$, where A^- is a new concept;*
- *If $C = \top$, then $\pi(C) = \top$;*
- *If $C = \bot$, then $\pi(C) = \bot$;*
- *If $C = E \sqcap D$ for concepts D, E, then $\pi(C) = \pi(E) \sqcap \pi(D)$;*
- *If $C = E \sqcup D$ for concepts D, E, then $\pi(C) = \pi(E) \sqcup \pi(D)$;*
- *If $C = \exists R.D$ for D a concept and R is a role, then $\pi(C) = \exists R.\pi(D)$;*
- *If $C = \forall R.D$ for D a concept and R is a role, then $\pi(C) = \forall R.\pi(D)$;*
- *If $C = \neg\neg D$ for a concept D, then $\pi(C) = \pi(D)$;*
- *If $C = \neg(E \sqcap D)$ for concepts D, E, then $\pi(C) = \pi(\neg E) \sqcup \pi(\neg D)$;*
- *If $C = \neg(E \sqcup D)$ for concepts D, E, then $\pi(C) = \pi(\neg E) \sqcap \pi(\neg D)$;*
- *If $C = \neg(\exists R.D)$ for D a concept and R is a role, then $\pi(C) = \forall R.\pi(\neg D)$;*
- *If $C = \neg(\forall R.D)$ for D a concept and R is a role, then $\pi(C) = \exists R.\pi(\neg D)$;*

Based on this, axioms are transformed as follows.

Definition 3 *(Axiom Transformations) For any ontology O, $\overline{O}$ is defined as the set $\{\overline{\alpha} \mid \alpha$ is an axiom of $O\}$, where $\overline{\alpha}$ is the transformation performed on each axiom defined as follows:*

- $\pi(\alpha) = \neg\pi(\neg C_1) \sqsubseteq \pi(C)_2$, *if* $\alpha = C_1 \mapsto C_2$;
- $\pi(\alpha) = \pi(C_1) \sqsubseteq \pi(C)_2$, *if* $\alpha = C_1 \sqsubseteq C_2$;
- $\pi(\alpha) = \{\pi(C_1) \sqsubseteq \pi(C)_2, \pi(\neg C_2) \sqsubseteq \pi(\neg C_1)\}$, *if* $\alpha = C_1 \rightarrow C_2$;.

- $\pi(C(a)) = \pi(C)(a), \pi(R)(a,b) = R(a,b)$,

where a, b are individuals, C_1, C_2, C are concepts, R a role.

We note two issues. First of all, the transformation algorithm is linear in the size of the ontology. Secondly, for any $\mathcal{ALC}$ ontology O, $\overline{O}$ is still an $\mathcal{ALC}$ ontology. Based on these two observations as well as the following theorem, we can see that paraconsistent reasoning of $\mathcal{ALC}$ can indeed be simulated on standard reasoners by means of the transformation just given. This also implies that paraconsistent reasoning in our paradigm is not more expensive than classical reasoning.

Theorem 1 *For any ontology O in $\mathcal{ALC}$ we have $O \models_4 \alpha$ if and only if $\pi(O) \models_2 \pi(\alpha)$, where $\models_2$ is the entailment in classical $\mathcal{ALC}$.*

In the rest of this section we prove theorem 1. For this, we first need some notations and definitions.

Definition 4 *(Decomposability) The four-valued semantics of $\mathcal{ALC}4$ is said to be decomposable into two-valued semantics of $\mathcal{ALC}$, if and only if for any concept C in an $\mathcal{ALC}4$ ontology $\mathcal{O}$, there are two concepts C_1,C_2 in a two-valued $\mathcal{ALC}$ ontology $\mathcal{O}'$ such that for any four-valued interpretation $\mathcal{I}$ of $\mathcal{O}$, there is a two-valued interpretation $\mathcal{I}'$ of $\mathcal{O}'$ which satisfies*

$$C^{\mathcal{I}} = \langle P, N\rangle \quad \textit{iff} \quad C_1^{\mathcal{I}'} = P, C_2^{\mathcal{I}'} = N,$$

where P, N, P_1, P_2, N_1 and N_2 are subsets of the domain of $\mathcal{I}$.

The decomposability of $\mathcal{ALC}4$ means that the four-valued semantics of a concept C can be divided into the two-valued semantics of two $\mathcal{ALC}$ concepts C_1, C_2. This is an essential property of four-valued DLs such that Theorem 1 holds.
To avoid any notational ambiguities between the original and the transformed languages, we will denote the original language by $\mathcal{L}$, for which $\mathcal{L} = \{C, R, a \mid C$ is a concept name, R is a role name, a is an individual$\}$. $\mathcal{A}(\mathcal{L})$ is the atomic concept set of $\mathcal{L}$. The transformed language is denoted as $\pi(\mathcal{L}) = \{\pi(C), \pi(\neg C), \pi(R), \pi(a) \mid C, R, a \in \mathcal{L}$, $\pi(C), \pi(\neg C)$ are the concept transformations of C and $\neg C$ respectively, $\pi(R)$ and $\pi(a)$ are renamed name of role R and of individual a, respectively$\}$.

Definition 5 *(Classical Induced Interpretation) Let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be a four-valued interpretation and $\pi(\mathcal{O})$ be the classical induced ontology of $\mathcal{O}$. The* classical induced interpretation *$\pi(\mathcal{I}) = (\Delta^{\pi(\mathcal{I})}, \cdot^{\pi(\mathcal{I})})$ of $\mathcal{I}$ is defined as follows:*

- *$\mathcal{I}$ and $\pi(\mathcal{I})$ have the same domain, i.e. $\Delta^{\pi(\mathcal{I})} = \Delta^{\mathcal{I}}$;*
- *$\mathcal{I}$ and $\pi(\mathcal{I})$ interpret instance names in the same way, i.e. $\pi(a)^{\pi(\mathcal{I})} = a^{\mathcal{I}}$;*
- *For any atomic concept A, if $A^{\mathcal{I}} = \langle P, Q\rangle$, then $(A^+)^{\pi(\mathcal{I})} = P$ and $(A^-)^{\pi(\mathcal{I})} = Q$;*
- *The semantics of complex concepts is obtained in the standard way.*

Definition 6 *(Four-valued Induced Interpretation) Let $\pi(\mathcal{I})$ be the interpretation of an $\mathcal{ALC}$ ontology $\mathcal{O}$. The* four-valued induced interpretation *$\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ of $\pi(\mathcal{I})$ is defined as follows:*

- *$\mathcal{I}$ and $\pi(\mathcal{I})$ have the same domain, i.e. $\Delta^{\pi(\mathcal{I})} = \Delta^{\mathcal{I}}$;*
- *$\mathcal{I}$ and $\pi(\mathcal{I})$ interpret instance names and roles in the same way, i.e. $\pi(a)^{\pi(\mathcal{I})} = a^{\mathcal{I}}, \pi(R)^{\pi(\mathcal{I})} = R^{\mathcal{I}}$;*
- *For any primitive concept A, if $(A^+)^{\pi(\mathcal{I})} = P, (A^-)^{\pi(\mathcal{I})} = Q$, then $A^{\mathcal{I}} = \langle P, Q\rangle$;*

- *The semantics of complex concepts is obtained according to Table 2.2.*

Now we prove the decomposability of $\mathcal{ALC}4$.

Lemma 1 *$\mathcal{ALC}4$ can be decomposed to two-valued semantics of $\mathcal{ALC}$.*

Proof 1 *Let $\mathcal{O}$ be an $\mathcal{ALC}4$ ontology and C be a concept. For any interpretation $\mathcal{I}$, we prove by induction on the concept structure that $C^{\mathcal{I}} = \langle P, N\rangle$ iff $\pi(C)^{\pi(\mathcal{I})} = P$ and $\pi(\neg C)^{\pi(\mathcal{I})} = N$, where $\pi(\mathcal{I})$ is the classical induced interpretation of $\mathcal{I}$.*

- *Case: C is an atomic concept A is easy by Definitions 6 and 5.*
- *Case: $C = \neg D$. So $\pi(C) = \pi(\neg D)$ and $\pi(\neg C) = \pi(D)$.*

  (Only If) Suppose $C^{\mathcal{I}} = \langle P, N\rangle$. Then $D^{\mathcal{I}} = \langle N, P\rangle$. By induction assumption, we know $\pi(D)^{\pi(\mathcal{I})} = N$ and $\pi(\neg D)^{\pi(\mathcal{I})} = P$. That is $\pi(\neg C)^{\pi(\mathcal{I})} = N$ and $\pi(C)^{\pi(\mathcal{I})} = P$.

  (If) Suppose $\pi(C)^{\pi(\mathcal{I})} = P, \pi(\neg C)^{\pi(\mathcal{I})} = N$. Then $\pi(D)^{\pi(\mathcal{I})} = N, \pi(\neg D)^{\pi(\mathcal{I})} = P$. By induction assumption, we know $D^{\mathcal{I}} = \langle N, P\rangle$. Through the semantics of negation, we know $C^{\mathcal{I}} = \langle P, N\rangle$.
- *Case: $C = D \sqcup E$. $\pi(C) = \pi(D) \sqcup \pi(E)$, and $\pi(\neg C) = \pi(\neg D) \sqcap \pi(\neg E)$.*

  (Only If) Suppose $C^{\mathcal{I}} = \langle P, N\rangle$,$D^{\mathcal{I}} = \langle P_1, N_1\rangle, E^{\mathcal{I}} = \langle P_2, N_2\rangle$. Then $P_1 \cup P_2 = P, N_1 \cap N_2 = N$. By induction assumption, we know $\pi(D)^{\pi(\mathcal{I})} = P_1, \pi(\neg D)^{\pi(\mathcal{I})} = N_1, \pi(E)^{\pi(\mathcal{I})} = P_2$, and $\pi(\neg E)^{\pi(\mathcal{I})} = N_2$. Therefore $\pi(C)^{\pi(\mathcal{I})} = \pi(D)^{\pi(\mathcal{I})} \cup \pi(E)^{\pi(I)} = P_1 \cup P_2 = P$ and $\pi(\neg C)^{\pi(\mathcal{I})} = \pi(\neg C)^{\pi(\mathcal{I})} \cap \pi(\neg E)^{\pi(\mathcal{I})} = N_1 \cap N_2 = N$.

  (If) Suppose $\pi(C)^{\pi(\mathcal{I})} = P, \pi(\neg C)^{\pi(\mathcal{I})} = N, \pi(D)^{\pi(I)} = P', \pi(\neg C)^{\pi(\mathcal{I})} = N'$, and $\pi(E)^{\pi(I)} = P'', \pi(\neg E)^{\pi(I)} = N''$. By the definition of the semantics, $P = P' \cup P''$ and $N = N' \cap N''$. By induction assumption, we know $D^{\mathcal{I}} = \langle P', N'\rangle$ and $E^{\mathcal{I}} = \langle P'', N''\rangle$. Therefore $C^{I} = \langle P' \cup P'', N' \cap N''\rangle = \langle P, N\rangle$ by definition of the semantics of $\mathcal{ALC}4$.
- *Case: $C = D \sqcap E$. The proposition holds likewise.*
- *Case: $C = \forall R.D$. $\pi(C) = \forall R.\pi(D)$ and $\pi(\neg C) = \exists R.\pi(\neg D)$,*

  (Only If) Suppose $C^{\mathcal{I}} = \langle P, N\rangle$ and $D^{\mathcal{I}} = \langle P_1, N_1\rangle$. By definition definition of the semantics, we know $P = \{x \mid \forall y, R(x,y) \Rightarrow y \in proj^{+}(D^{\mathcal{I}})\}$ and $N = \{x \mid \exists y, R(x,y) \wedge y \in proj^{-}(D^{\mathcal{I}})\}$. By induction assumption, we know $\pi(D)^{\pi(\mathcal{I})} = P_1$ and $\pi(\neg D)^{\pi(\mathcal{I})} = N_1$. Therefore, $N_1 = proj^{-}(D^{\mathcal{I}})$. (Note that $P_1 = proj^{+}(D^{\mathcal{I}})$).

$$\begin{aligned}
\pi(C)^{\pi(\mathcal{I})} &= (\forall R.\pi(D)^{\pi(\mathcal{I})} = \{x \mid \forall y, R(x,y) \Rightarrow y \in (\pi(D)^{\pi(\mathcal{I})}\} \\
&= \{x \mid \forall y, R(x,y) \Rightarrow y \in P_1\} = P, \\
\pi(\neg C)^{\pi(\mathcal{I})} &= (\exists R.\pi(\neg D))^{\pi(\mathcal{I})} = \{x \mid \exists y, R(x,y) \wedge y \in (\pi(\neg D))^{\pi(\mathcal{I})}\} \\
&= \{x \mid \exists y, R(x,y) \wedge y \in N_1\} = N.
\end{aligned}$$

  (If) Suppose $\pi(C)^{\pi(\mathcal{I})} = P, \pi(\neg C)^{\pi(\mathcal{I})} = N, \pi(D)^{\pi(\mathcal{I})} = P', \pi(\neg D)^{\pi(\mathcal{I})} = N'$. By the definition of the semantics,

$$\begin{aligned}
P &= \pi(C)^{\pi(\mathcal{I})} = (\forall R.\pi(D)^{\pi(\mathcal{I})} = \{x \mid \forall y, R(x,y) \Rightarrow y \in P'\}, \\
N &= \pi(\neg C)^{\pi(\mathcal{I})} = (\exists R.\pi(\neg D))^{\pi(\mathcal{I})} = \{x \mid \exists y, R(x,y) \wedge y \in N'\}.
\end{aligned}$$

  By induction assumption, we know $D^{\mathcal{I}} = \langle P', N'\rangle$. Furthermore, by the semantics of $\mathcal{ALC}4$, we know

$$C^{\mathcal{I}} = \langle \{x \mid \forall y, R(x,y) \Rightarrow y \in P'\} \quad \textit{and} \quad \{x \mid \exists y, R(x,y) \wedge y \in N'\}\rangle = \langle P, N\rangle.$$

- *Case $C = \exists R.D$, the lemma holds likewise.*

In all, let $C_1 = \pi(C), C_2 = \pi(\neg C)$, we see that for any concept C, $C^{\mathcal{I}} = \langle P, N\rangle$ iff $C_1^{\pi(\mathcal{I})} = P$ and $C_2^{\pi(\mathcal{I})} = N$. □

Now we turn to prove theorem 1.

Proof 2 *(OF THEOREM 1) (Necessity) For any interpretation I of $\mathcal{O}$, let $\pi(\mathcal{I})$ be the classical induced interpretation of $\mathcal{I}$. According to the relationship between $\pi(\mathcal{O})$ and $\mathcal{O}$, for any axiom in $\pi(\mathcal{O})$ of the form $\neg\pi(\neg C) \sqsubseteq \pi(D) \in \pi(\mathcal{O})$ we have $C \mapsto D \in \mathcal{O}$. Suppose $C^{\mathcal{I}} = \langle P_1, N_1\rangle$ and $D^{\mathcal{I}} = \langle P_2, N_2\rangle$. By lemma 1, $\pi(\neg C)^{\pi(\mathcal{I})} = N_1$ and $\pi(D)^{\pi(\mathcal{I})} = P_2$ hold. Then, $(\neg\pi\neg C)^{\pi(\mathcal{I})} = \Delta^{\pi(\mathcal{I})} \setminus N_1 = \Delta^{\mathcal{I}} \setminus N_1$. $\mathcal{I}$ satisfies $C \mapsto D$, so $\Delta^{\mathcal{I}} \setminus N_1 \subseteq P_2$. Therefore, $(\neg\pi(\neg C))^{\pi(\mathcal{I})} \subseteq \pi(D)^{\pi(\mathcal{I})}$, that is, $\pi(\mathcal{I})$ satisfies $\neg\pi(\neg C) \sqsubseteq \pi(D)$.*
For any axiom in $\pi(\mathcal{O})$ of the form $\pi(C) \sqsubseteq \pi(D) \in \pi(\mathcal{O})$ with $\pi(\neg C) \sqsubseteq \pi(\neg D) \notin \pi(\mathcal{O})$ we have $C \sqsubset D \in \mathcal{O}$. Suppose $C^{\mathcal{I}} = \langle P_1, N_1\rangle$ and $D^{\mathcal{I}} = \langle P_2, N_2\rangle$. By lemma 1 we have $\pi(C)^{\pi(\mathcal{I})} = P_1$ and $\pi(D)^{\pi(\mathcal{I})} = P_2$. $\mathcal{I}$ satisfies $C \sqsubset D$. Therefore, $P_1 = proj^+(C^{\mathcal{I}}) \subseteq proj^+(D^{\mathcal{I}}) = P_2$, that is $\pi(\mathcal{I})$ satisfies $\pi(C) \sqsubseteq \pi(D)$.
For any axiom in $\pi(O)$ of the form $\{\pi(C) \sqsubseteq \pi(D), \pi(\neg D) \sqsubseteq \pi(\neg C)\} \subseteq \pi(\mathcal{O})$ we have $C \to D \in \mathcal{O}$. Suppose $C^{\mathcal{I}} = \langle P_1, N_1\rangle, D^{\mathcal{I}} = \langle P_2, N_2\rangle$. By lemma 1 we have $\pi(C)^{\pi(\mathcal{I})} = P_1, \pi(D)^{\pi(\mathcal{I})} = P_2, \pi(\neg C)^{\pi(\mathcal{I})} = N_1$, and $\pi(\neg D)^{\pi(\mathcal{I})} = N_2$. $\mathcal{I}$ satisfies $C \to D$. Therefore, $P_1 = proj^+(C^{\mathcal{I}}) \subseteq proj^+(D^{\mathcal{I}}) = P_2, N_2 = proj^-(D^{\mathcal{I}}) \subseteq proj^-(C^{\mathcal{I}}) = N_1$, that is $\pi(\mathcal{I})$ satisfies $\{\pi(C) \sqsubseteq \pi(D), \pi(\neg D) \sqsubseteq \pi(\neg C)\}$.
For any assertion of the form $\pi(a)\!:\!\pi(C)$, we have that $a\!:\!C$ belongs to $\mathcal{O}$. Suppose $C^{\mathcal{I}} = \langle P, N\rangle$ and $a^{\mathcal{I}} = \delta_0 \in \Delta^{\mathcal{I}}$, then $(\pi C)^{\pi(\mathcal{I})} = P$ and $\pi(a)^{\pi(\mathcal{I})} = \delta_0$. Because $\mathcal{I}$ satisfies $a\!:\!C$ we have $\delta_0 \in P$, that is $\pi(\mathcal{I})$ satisfies $\pi(a)\!:\!\pi(C)$. It is similar for statements of the form $R(a, b)$.
(Sufficiency) For any interpretation $\pi(\mathcal{I}) = (\Delta^{\pi(\mathcal{I})}, \cdot^{\pi(\mathcal{I})})$ of $\pi(O)$, let $\mathcal{I}$ be the four-valued semantics of $\pi(\mathcal{I})$. Similarly, we can prove the proposition to be right. □

2.4 Paraconsistent Semantics for Expressive DLs

In this section, we study how to extend four-valued semantics to $\mathcal{SHIQ}$.
For the conflicting assertion set $\{\geq (n+1)R.C(a), \leq nR.C(a)\}$, intuitively, it is caused by the contradiction that there should be less than n different individuals related to a via the R relation, and also there should be more than $n+1$ different individuals related to a via R. That is, the contradiction is from the set of individuals of concept C which relate a via R. By this idea, we extend the four-valued semantics to the constructors for number restrictions in Table 2.5. We remark that the semantics of roles is just the classical semantics. So the semantics for role inclusion and transitive role axiom are still classical.

Table 2.5: Four-valued Semantics Extension to Number Restrictions and Nominals

Constructor	Semantics
$\geq nR.C$	$\langle\{x \mid \#(y.(x,y) \in R^{\mathcal{I}} \wedge y \in proj^+(C^{\mathcal{I}})) \geq n\}$, $\{x \mid \#(y.(x,y) \in R^{\mathcal{I}} \wedge y \notin proj^-(C^{\mathcal{I}})) < n\}\rangle$
$\leq nR.C$	$\langle\{x \mid \#(y.(x,y) \in R^{\mathcal{I}} \wedge y \notin proj^-(C^{\mathcal{I}})) \leq n\}$, $\{x \mid \#(y.(x,y) \in R^{\mathcal{I}} \wedge y \in proj^+(C^{\mathcal{I}})) > n\}\rangle$

Example 1 *Consider $\{\geq 2hasStu.PhD(Green), \leq 1hasStu.PhD(Green)\}$ which says the conflicting facts that $Green$ has at least two and at most one PhD student. Consider a 4-interpretation: $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ where $\Delta^{\mathcal{I}} = \{a_1, a_2, b_1, b_2, Green\}$, $PhD^{\mathcal{I}} = \langle\{a_1, b_1\}, \{b_1, b_2, a_2\}\rangle$, and $hasStu^{\mathcal{I}} = \{(Green, a_1),$*

$(Green, a_2)$, $(Green, b_1)$, $(Green, b_2)\}$. *According to Table 2.5, we can see that* $\mathcal{I}$ *is a 4-model because* $(\geq 2hasStu.PhD(Green))^{\mathcal{I}}$ = $(\leq 1hasStu.PhD(Green))^{\mathcal{I}} = B$ *by checking*

$$Green \in \{x \mid \#(y.(x,y) \in hasStu^{\mathcal{I}} \wedge y \in \textit{proj}^{+}(PhD^{\mathcal{I}})) \geq 2\},$$
$$Green \in \{x \mid \#(y.(x,y) \in hasStu^{\mathcal{I}} \wedge y \notin \textit{proj}^{-}(PhD^{\mathcal{I}})) < 2\}.$$

That is, the conflicting assertions are assigned the contradictory truth value B *under their 4-model* $\mathcal{I}$*.*

For the extended four-valued semantics defined in Table 2.5, we have following properties hold as under classical semantics.

Proposition 4 *Let* C *be a concept and* R *be an object role name. For any four-valued interpretation* I *defined satisfying Table 2.5, we have*

$$(\neg(\leq nR.C))^{\mathcal{I}} =_4 (> nR.C)^{\mathcal{I}} \quad \textit{and} \quad (\neg(\geq nR.C))^{\mathcal{I}} =_4 (< nR.C)^{\mathcal{I}}.$$

Proposition 5 *Let* C *be a concept and* R *be an object role name. For any four-valued interpretation* $\mathcal{I}$ *defined satisfying Table 2.5, we have*

$$(\exists R.C)^{\mathcal{I}} =_4 (\geq 1R.C)^{\mathcal{I}} \quad \textit{and} \quad (\forall R.C)^{\mathcal{I}} =_4 (< 1R.\neg C)^{\mathcal{I}}.$$

Proposition 4 and Proposition 5 show that many intuitive relations between different concept constructors still hold under the four-valued semantics, which is one of nice properties of our four-valued semantics for handling inconsistency.

Next proposition shows that our definition of four-valued semantics for $\mathcal{SHIQ}$ is enough to handle inconsistencies in an $\mathcal{SHIQ}$ knowledge base.

Definition 7 *Given a knowledge base* O*, the satisfiable form of* O*, written SF*(O)*, is a knowledge base obtained by replacing each occurrence of* $\bot$ *in* O *with* $A_{new} \sqcap \neg A_{new}$*, and replacing each occurrence of* $\top$ *in* (O) *with with* $A_{new} \sqcup \neg A_{new}$*, where* A_{new} *is a new atomic concept.*

Proposition 6 *For any* $\mathcal{SHIQ}$ *knowledge base* O*, SF*(O) *always has at least one 4-valued model, where SF*$(\cdot)$ *operator is defined in Definition 7.*

Note that unqualified number restrictions, $\geq n.R$ and $\leq n.R$ are special forms of number restrictions because of the equations $\leq n.R =_2 \leq nR.\top$ and $\geq n.R =_2 \geq nR.\top$. However, if we defined the four-valued semantics of $\leq n.R (\geq n.R)$ by the four-valued semantics of $\leq nR.\top (\geq nR.\top)$ defined in Table 2.5 and Table 2.2, we would find that $\{\leq n.R(a), \geq n+1.R(a)\}$ is still an unsatisfiable set. This is because $\#(y.(a,y) \in \textit{proj}(R^{\mathcal{I}}) \wedge y \in \textit{proj}^{+}(\top^{\mathcal{I}})) \geq n+1$ and $\#(y.(a,y) \in \textit{proj}(R^{\mathcal{I}}) \wedge y \notin \textit{proj}^{-}(\top^{\mathcal{I}})) \leq n$ cannot hold simultaneously since $\top^{\mathcal{I}} = \langle \Delta^{\mathcal{I}}, \varnothing \rangle$.

To address this problem, we also adopt the *substitution* defined by Definition 7. By substituting $\top$ by $A_{new} \sqcup \neg A_{new}$ in $\geq (n+1)R.\top$ and $\leq nR.\top$, we can see that $\{\leq n.R(a), \geq n+1.R(a)\}$ has a four-valued model with $\Delta^{\mathcal{I}} = \{a, b_1, ..., b_{n+1}\}, (a, b_i) \in R^{\mathcal{I}}$ for $1 \leq i \leq n+1$, *and* $A^{\mathcal{I}}_{new} = \langle \Delta^{\mathcal{I}}, \Delta^{\mathcal{I}} \rangle$. By doing this, we get a four-valued model $\mathcal{I}$ which pushes the contraction onto the new atomic concept A_{new}.

Next we study how to extend the reduction algorithm to the case of four-valued semantics of $\mathcal{SHIQ}$.

Definition 8 *For any given concept* C*, its transformation* $\pi(C)$ *is the concept obtained from* C *by the following inductively defined transformation.*

- *If* $C =\geq nR.D$ *for* D *a concept and* R *a role, then* $\pi(C) =\geq nR.\pi(D)$*;*
- *If* $C =\leq nR.D$ *for* D *a concept and* R *a role, then* $\pi(C) =\leq nR.\neg\pi(\neg D)$*;*

- *If* $C = \neg(\geq nR.D)$ *for* D *a concept and* R *a role, then* $\pi(C) =< nR.\neg\pi(\neg D)$*;*
- *If* $C = \neg(\leq nR.D)$ *for* D *a concept and* R *a role, then* $\pi(C) => nR.\pi(D)$*;*

Regarding both the extension of number restrictions and of nominals, the following theorem holds, which lays the theoretical foundation for the algorithm of four-valued semantics for expressive DLs.

Theorem 2 *(Theorem 1 extended) For any ontology* O *in* $\mathcal{SHIQ}$*,* O *is 4-valued unsatisfiable if and only if* $\pi(O)$ *is unsatisfiable under the classical semantics of* $\mathcal{SHIQ}$*.*

2.5 Implementation of a Paraconsistent Reasoning Algorithm

2.5.1 Functionality of the Inconsistancy Handler plug-in

The *Inconsistency Handler* is a plug-in that allows for the Neon Toolkit to transform inconsistent OWL ontologies into non-contradictory knowledge bases. To this end the ontology is considered to be based on 4-valued logic and and is then transformed. If afterwards the the 2-valued reasoning of the Toolkit is applied to the new knowledge base, the results will correspond to a 4-valued reasoning under the original ontology. In this manner the Neon Toolkit can work using 4-valued logic, without the need to develop a new inference machine. A link to the description of the plugin is provided at http://www.neon-toolkit.org/wiki/index.php/ParOWL.

2.5.2 The program logic of the Inconsistency Handler

On execution of the transformation the plug-in queries all axioms of the knowledge base and translates them individually. The axioms are implicitly split into three categories.

- **No inclusion**: Some axioms cannot be described as an inclusion, since, for example, they might represent a concept assertion or simply the declaration of a class. These are easier to handle during the transformation than other axioms.
- **Irrelevant inclusion**: These are inclusions which do not have to be treated as strong, material or internal inclusions, for example role inclusions. They can be preserved in their current form, only their descriptions have to be transformed. Furthermore this category comprises inclusions which do not contain any concepts in their subclass except $\top$ or $\bot$. Since these are preserved in internal and material inclusions, these axioms, too, need only a translation of their descriptions. Only in case of a strong inclusion one of the resulting axioms needs to see $\top$ replaced with $\bot$, and vice versa.
- **Relevant inclusion**: All axioms which are not represented by either of the two classes just described have to be considered internal, strong or material inclusions, and must be transformed accordingly. It is important that these axioms, regardless of their type before the translation, result in one or more axioms of the type `SubClassOf`.

All transformed axioms are included in their respective ontology while the original axioms are removed.

Launch of the plug-in

The actual starting point of the plug-in within the program logic is the class `Action`, which implements the interface `IWorkbenchWindowActionDelegate`. The latter allows definition of the initializing method `run(IAction action)` which is first to be executed on activation of the Inconsistency Handler. Within, an `ISelectionService` checks which element is currently selected inside the window. If this element is an ontology project, a `TranslationCore`generated - it's constructor receiving the name of the project - and it's method `startTrans()` is invoked. If anything else is selected the procedure is aborted and a message is displayed.

The class `TranslationCore` controls the transformation of an entire project and it's ontologies. `startTrans()` returns the ontologies of the project which was used to initialize the class. The user decides if all inconsistent ontologies are to be processed, or if he wants to confirm processing for each ontology individually. Either way an object of the type `Reasoner` is used to check the satisfiability of the individual ontologies. In case all ontologies are processed automatically this leads to a decision whether the respective ontology is to be transformed. If transformation of each individual ontology needs to be confirmed by the user, `Reasoner` serves to inform him about the ontologies consistency.

Since the Reasoner only works for $\mathcal{SHIQ}$(D), it is possible to secure at this point that knowledge bases are only transformed if their axioms are included in the transformation, that is, if they are in $\mathcal{SHIQ}$(D). To this end any KAON2-exception terminating the process is intercepted, and a message is displayed.

For each ontology that needs to be translated, the method

```
void translation (boolean shortTrans, Ontology ontology)
```

is invoked, which controls the acutal transformation process. The parameter `shortTrans` decides whether all inclusions of the ontology are handled in the same manner, or if manner of handling has to be decided on an individual basis. It's parameters are set using a dialogue defined in the class `InclusionDialog`, which provides all methods used to control dialogues concerning the handling of inclusions.

Transforming an ontology

The method `translation` controls the transformatory process of a single ontology. For this, three `transformer`-classes are instantiated, one each of the types `Strong`, `Internal` and `Material`. These are responsible for the translation of the individual axioms, whereas each handles inclusions according to its type. Each OWL axiom within the ontology is queried and checked for its type, after which it is cast into an object of that type. The object is now handed to one of the three `transformer`-instances as a parameter of the method

```
Set<Axiom> transform (OWLAxiom axiom)
```

The instance then translates the axiom and returns a new set of axioms. Which instance is adressed depends on the variable iType, which is either set in advance for all axioms, or individually for each one - this depends on the boolean parameter described earlier. The value of `iType` is set through a dialogue which is itself defined in the class `InclusionDialog`. If this value is set for each axiom individually, the name of the axiom can be handed to the method that starts the dialogue, so as to make clear to the user which axiom he is currently making a decision on.

During the entire process a list of so-called `OntologyChangeEvent`s is filled. Each accessed axiom from the original ontology is added to this list as `ChangeType.REMOVE` after the method `transform` has been invoked. Those axioms that have been transformed and returned are subsequently added as `ChangeType.ADD`. Finally the `OntologyChangeEvent`s are applied to the ontology, which results in replacement of all original axioms by the transformed ones.

Transforming an axiom

Transformation of individual axioms is handled by the abstract class `Transformer`. This class should be considered the heart of the Inconsistency Handler, as it and its subclasses are conducting the actual transformation. To this end it contains the method `transform`, which through overloading realizes the already discussed categorization of the axioms.

This method is specified differently for each type of OWL axiom it can be handed. In case of statements that match one or more relevant inclusions, their descriptions are extracted and extended, so they can be interpreted as one or more classic inclusions and be processed accordingly. For example, if `transform` is being handed the axiom

```
[DisjointClasses Plant [or Human Animal]]
```

this corresponds to an inclusion of the form

$$\textit{Plant} \sqsubseteq \neg(\textit{Human} \sqcup \textit{Animal})$$

Consequently, the descriptions

```
Plant
[not [or Human Animal]]
```

will be derived and further processed.
The descriptions modified in this way are handed to the abstract method

```
Set<Axiom> finalTrans (StringBuffer part, StringBuffer whole)
```

It is realised in classes `Strong`, `Material` and `Internal` (which inherit from `Transformer`) in a way so they will transform the descriptions and create axioms of the type `SubClassOf`. Concepts are negated according to transformation directives that apply to the different types of inclusions.
In case `transform` is handed axioms that constitute an inclusion, but do not need to be processed as strong, material or internal transformations, these axioms are passed on directly to the method

```
Set<Axiom> propInc (OWLAxiom axiom)
```

, too, extracts the contained descriptions. However, it will recompose them as an axiom of the original type after the transformation. Note that axioms which define domain or range of properties constitute an exception. Put in description logic they correspond to:

$\top \sqsubseteq \forall R.C$ (Range of R is C)
$\exists R.\top \sqsubseteq C$ (Domain of R is C)

Transformation according to the different kinds of inclusions yields:

- for Internal:

$\overline{\top} \sqsubseteq \overline{\forall R.C}$ is corresponding to $\top \sqsubseteq \forall R.\overline{C}$
$\overline{\exists R.\top} \sqsubseteq \overline{C}$ is corresponding to $\exists R.\top \sqsubseteq \overline{C}$

- for Material:

$\neg\overline{\neg\top} \sqsubseteq \overline{\forall R.C}$ is corresponding to $\top \sqsubseteq \forall R.\overline{C}$
$\neg\overline{\neg\exists R.\top} \sqsubseteq \overline{C}$ is corresponding to $\exists R.\top \sqsubseteq \overline{C}$

- for Strong:

$\{\overline{\top} \sqsubseteq \overline{\forall R.C},\ \overline{\neg\top} \sqsubseteq \overline{\neg\forall R.C}\}$ is corresponding to $\{\top \sqsubseteq \forall R.\overline{C},\ \bot \sqsubseteq \exists R.\overline{\neg C}\}$
$\{\overline{\exists R.\top} \sqsubseteq \overline{C},\ \overline{\neg\exists R.\top} \sqsubseteq \overline{\neg C}\}$ is corresponding to $\{\exists R.\top \sqsubseteq \overline{C},\ \forall R.\bot \sqsubseteq \overline{\neg C}\}$

It shows that in any case the result will be an inclusion which again represents a range or a domain. Solely concept C has to be transformed. Thus these axioms can be handed to the method `propInc`, as well. Since an additional axiom is created in case of a strong inclusion, the method inside the class `Strong` is shadowed and supplemented with additional mechanisms. These are the mechansisms responsible for the abovementioned additional statements in case of domain and range axioms.

The described approach for inclusions that are irrelevant to transformation has been chosen to allow type preservation for as many axioms as possible. Among others, this offers the advantage of allowing full use of the visualization methods provided by the Neon Toolkit. If, instead, most axioms were transformed into axioms of the type `SubClassOf`, visualization would be highly restricted.

Finally there is one version of the `transform` method that accepts general OWL axioms as parameter, and is therefore called upon for all axioms that are not treated by the previously described variants of the method. This category mainly comprises all statements which do not represent an inclusion, such as concept assertions or simply concept declarations. These are preserved in form, with only their descriptions transformed. This version of `transform` will also catch and process axioms that are outside $\mathcal{SHIQ}$(D), in case the safety mechanisms of `startTrans` described earlier should fail to prevent that.

Also, please note that for some axioms there is a version for data types, as well. For example, `ObjectPropertyRange` describes the range of properties that have been defined for "'normal"' concepts, while `DataPropertyRange` defines the range of properties that refer to data types. Regarding transformation however, this issue can be neglected, since both versions can always be transformed in the same way. Thus data types and concepts can be treated equivalently in matters of transformation.

Transforming a description

In the KAON2 data model, individual concepts are represented as descriptions. They can occur atomically as URIs, or complex, assembled by a constructor. Transformation of these descriptions is handled through the method

```
Description rename(StringBuffer desc, boolean not)
```

It is defined within the class `Transformer`, and is employed for transformation of individual inclusions by `propInc` as well as by the three variants of `finalTrans`. Furthermore it is used by `transform`, to generate descriptions for those axioms that are not inclusions.

The method `rename` accepts a single description as a stringbuffer. In a first step the type of a description is checked by

```
descType check(StringBuffer descrip)
```

which is also defined in `Transformer`. It scans the beginning of the description for keywords that will reveal its type. In regard to the description logic this scan corresponds to an assessment of the kind of the outermost constructor of the statement that needs to be considered first in the course of transformation. For example, the description

```
[and [or Cake Cookies] [or Koffee Tea]]
```

would return "'`and`"' as type of the description, since in

$$((\mathit{Cake} \sqcup \mathit{Cookies}) \sqcap (\mathit{Coffee} \sqcup \mathit{Tea}))$$

the conjunction has to be considered first.

If the `check` method receives an atomic concept it will return '`simple`"' as a type. Another possible result of the check can be "'`unknown`"', in case the stringbuffer the method has been handed does not correspond to any predefined type.

Depending on the value received this way by the rename method, the description will be handled differently. There are two variants for each value, the decision between them depending on the boolean variable "'`not`"'. On an intuitive level this variable decides whether the concept is currently negated. If `not = false`, an atomic concept - that is, an URI - will be returned unmodified. However if `not = true`, a URI will be modified. This corresponds to the transformation rule

$\overline{A}$ transformed into A^+
$\overline{\neg A}$ transformed into A^-

where A is an atomic concept. The non-negated concept is not renamed, since renaming only the negated concepts is sufficient to ensure distinction. Modifying the URI is done by prefixing the string "'`not_`"' to the concept identifier, while `guessNamespaceEnd` prevents the namespace from being influenced.

Descriptions that represent complex concepts are split by the method

```
String[] splitter(String input, int argCount)
```

It contains the description as a string and returns a string array that contains the type of the description among its first elements. This type in turn corresponds to the outermost constructor of the concept. The remaining elements of the array contain the descriptions that are connected through that constructor. Please note that these descriptions may again be complex.

The constructors obtained in this way are modified according to the transformation rules, after which they are once again merged with their respective descriptions. The latter, however, are first processed recursively by `rename`, unless they are properties. Consider, for example, the following description being processed by `rename` (with the corresponding concept given in description logic):

rename([not [some drive [or Car Bicycle]]], *false*)
$\overline{\neg\exists drive.(Car \sqcup Bicycle)}$

In this case "'`not`"' would be cut off through use of method `splitter`, after which `rename` would be once more invoked, this time with the boolean variable set to *true*:

rename([some drive [or Car Bicycle]], *true*)
$\overline{\neg\exists drive.(Car \sqcup Bicycle)}$

Now `splitter` will decompose the description into three parts, with the property "'`drive`"' unmodified and, due to the negation, "'`some`"' replaced by "'`all`"'. Again, the remainder of the description is handed to `rename`, with the boolean variable unmofidfied:

[all drive *rename*([or Car Fahrrad], *true*)]
$\forall drive.\overline{\neg(Car \sqcup Bicycle)}$

During this invocation the disjunction becomes a conjunction and `rename` is applied to the remaining atomic concepts:

[all drive [and *rename*(Car, *t*) *rename*(Bicycle, *t*)]]
$\forall drive.(\overline{\neg Car} \sqcap \overline{\neg Bicycle})$

In the course of the two final invocations the renaming is carried out, which concludes the transformatory process for this description, and the recursive calls are traced back:

```
[all drive [and not_Car not_Bicycle]]
```
$\forall drive.(Car^- \sqcap Bicycle^-)$

In this way `rename` contains rules for every type of description. It is thus possible, through recursive calls, to process any given combination. If a specific type is not recognized and "'`unknown`"' is given as parameter, a message for the user is displayed accordingly and the concept is preserved as it is. However, this only applies to unambiguously complex descriptions, that is, if an unknown keyword is encountered.

Also, `rename` is quite fail-safe. If it is handed URIs describing properties or individuals, these are rated as `simple` type and returned unprocessed. This even applies to simple text strings, which bear no actual meaning in the knowledge base. Given the boolean variable is initialised with $false$, the method could even be applied to values that are only known at run time, although this is not necessary in terms of implementation.

2.5.3 Verification

The verification resulted from testing 30 different ontologies. Each knowledge base was transformed from its original form whereas all inclusions of transformation were perceived as strong, material and internal inclusion in each case. Ten of thirty ontologies were inconsistent. The twenty remaining ontologies were consistent.

After every transformation the remaining ontology was consistent which means that existent inconsistencies were successfully removed and no additional inconsistencies were created. This points out that the transformation was transcribed correctly.

The number of axioms in the tested ontologies was between 23 and 12656. The number of axioms in the transformed knowledge bases was between 29 and 24696. The number of axioms in the transformed ontologies was equal or higher than the number in the original ontologies. Furthermore the number of added axioms from internal and material transformation was always identical whereas less additional axioms were created here than in the strong alternative. But it seemed there was no correlation between the number of added axioms of strong transformation and the number of the other transformations. That is to say that the number of axioms after another transformation cannot be educed from the quantity of axioms after a strong transformation. There is simply an upper limit given by the number of axioms after the strong transformation.

The reason lies in the definition of the transformation. Whereas there are two axioms replacing the strong inclusions, there is just one axiom replacing the other inclusions. The reason for the change of number in the material and internal alternatives is that some axioms of the KAON2 data model are expressed by several inclusions. For example every `EquivalentClasses` axiom results in two `SubClassOf` axioms. These are doubled again in the strong alternative. Furthermore a correlation is not possible because there are axioms which do not result in any additional axioms, for example `ClassMember` which represents a concept assertion.

Summarized there are:

- axioms which never create other statements
- inclusions which create other inclusions just in the strong alternative
- axioms which always create other statements, but twice as much in the strong alternative

So in the best case no additional axioms are generated. But because some axioms result in any number of new axioms - for example `DisjointUnion` - no conclusions can be drawn about the worst case. But basically the strong transformation has the capability to create twice the amount of new axioms than the other transformation alternatives. In the conducted test the number of axioms in the strong transformation increased by a factor of 2.8 in the other alternatives by a factor of 1.6, but with big variations. There were ontologies which showed no difference in the number of axioms after an internal or material transformation. Other knowledge bases however resulted in twice the amount of axioms and nearly increased by a factor of 4 in the strong alternative. So it depends on the individual ontology to what extent the number of its axioms increases in the different transformations.

The increase of the number of axioms in the transformed ontologies matters because conclusions from large knowledge bases are more expansive in terms of computing time and processing power. Because the complexity of $\mathcal{SHIQ}$ is in EXPTIME, the complexity increases not only linear to the number of axioms. This means that the ability to draw a conclusion from an inconsistent knowledge base is probably paid by considerably increased complexity.

Chapter 3

Bilattice based reasoning with Description Logics

In this chapter, we generalize $\mathcal{ALC}_4$ towards the description logic SROIQ, which serves as the foundation of the upcoming Web Ontology language OWL2 and towards general bilattices with more than four truth values. Finally, we demonstrate the use of the extension by defining bilattices suitable for trust based reasoning.

We introduce logical bilattices in Section 3.1 and bilattice based reasoning with description logics in section 3.1.2. In Section 3.2 we use the extension for inferring trust in axioms derived from multiple, differently trustworthy data sources. Finally, in Section 3.2, we consider applications of bilattice-based reasoning.

3.1 Reasoning Based on Logical Bilattices

We will now discuss, how paraconsistent reasoning can be extended to more than four truth values. In the additional truth values, we will reflect additional dimensions of knowledge such as trust and uncertainty. In this deliverable, we focus on trust.

Most logic programming paradigms, including classical logic programming, stable model and well founded semantics, and fuzzy logics can be formalized based on bilattices of truth values and fixpoints of a direct consequence operator on such a bilattice. A bilattice is an algebraic structure, which encodes the semantics of logical connectives in multi valued logics. Even though the exact (parts of) bilattices used in a particular logical formalism differ, they provide a nice unifying framework, as discussed for example in [AA96]. Therefore, if we build our extension into this foundational layer, it will directly be available in many different formalisms. Furthermore, we will show how to extend description logics (which are not in general based on a fixpoint operator) can be extended towards logical bilattices.

A logical bilattice [Gin92] is a set of truth values, on which two partial orders are defined, which we call the truth order $\leq_t$ and the knowledge order $\leq_k$. Both $\leq_t$ and $\leq_k$ are complete lattices, i.e. they have a maximal and a minimal element and every two elements have exactly one supremum and infimum.

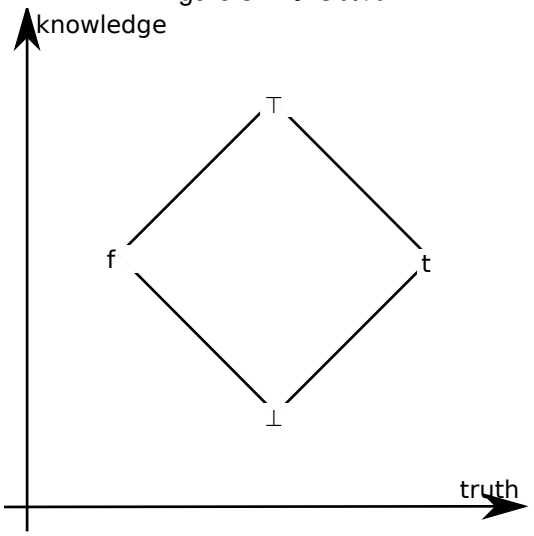
In logical bilattices, the operators $\vee$ and $\wedge$ are defined as supremum and infimum wrt. $\leq_t$. Analogously join ($\oplus$) and meet ($\otimes$) are defined as supremum and infimum wrt. $\leq_k$. As a result, we have multiple distributive and commutative laws, which all hold. Negation ($\neg$) simply is an inversion of the truth order. Hence, we can also define material implication ($a \rightarrow b = \neg a \vee b$) as usual.

The smallest non trivial logical bilattice is $\mathcal{FOUR}$, shown in figure 3.1. In addition to the truth values t and f, $\mathcal{FOUR}$ includes $\top$ and $\bot$. $\bot$ means "unknown", i.e. a fact is neither true or false. $\top$ means "overspecified" or "inconsistent", i.e. a fact is both true and false.

In traditional, two valued logic programming without negation, only t and f would be allowed as truth values. In contrast, e.g. the stable model semantics, allows to use $\top$ and $\bot$. In this case, multiple stable models are possible. For example, we might have a program with three clauses:

$man(bob) \leftarrow person(bob), \neg woman(bob).$

Figure 3.1: $\mathcal{FOUR}$



$woman(bob) \leftarrow person(bob), \neg man(bob).$
$person(bob).$
Using default f, we might infer both $man(bob) \wedge \neg woman(bob)$ and $woman(bob) \wedge \neg man(bob)$. While in two valued logics we would not be able find a model, in four values, we could assign truth values $t \oplus f = \top$ and $t \otimes f = \bot$. In fact, both would be allowed under the stable model semantics, resulting in multiple models for a single program.

The well founded semantics distinguishes one of these models — the minimal one, which is guaranteed to always exist and only uses t, f, and $\bot$. In a similar way, other formalisms can be expressed in this framework as well. Particularly, we can also formalize open world based reasoning, using $\bot$ instead of f as default value. We refer the interesting reader to the very good overview in [Fit02].

3.1.1 Obtaining bilattices

Ginsberg [Gin92] describes how we can obtain a logical bilattice: Given two distributive lattices $\mathcal{L}_1$ and $\mathcal{L}_2$, create a bilattice $\mathcal{L}$, where the nodes have values from $\mathcal{L}_1 \times \mathcal{L}_2$, such that the following orders hold:

- $\langle a, b\rangle \leq_k \langle x, y\rangle$ iff $a \leq_{\mathcal{L}_1} x \wedge b \leq_{\mathcal{L}_2} y$ and
- $\langle a, b\rangle \leq_t \langle x, y\rangle$ iff $a \leq_{\mathcal{L}_1} x \wedge y \leq_{\mathcal{L}_2} b$

If $\mathcal{L}_1$ and $\mathcal{L}_2$ are infinitely distributive— that means distributive and commutative laws hold for infinite combinations of the operators $\vee_1, \wedge_1$ and $\vee_2, \wedge_2$ respectively — then $\mathcal{L}$ will be as well, i.e. arbitrary combinations of $\vee, \wedge, \oplus,$ and$\otimes$ are possible in the resulting bilattice. In other words this means the language of logical connectives over bilattices is complete.

As an example, we model two levels of certainty for situation, where knowledge is cached. In such a situation, we may have loca information, which is certain, and cached information, which is possibly outdated. Such cached information will be assigned a special truth value. Moreover, during reasoning, we want to infer ho trustworthy the inferred informaiton can be.

For modeling this situation, generate a bilattice with two levels of truth values: certainly known truth values (subscripts k) and almost certain truth values (subscripts c). We use $\mathcal{L}_1 = \mathcal{L}_2 = t_k > t_c > f_c > f_k$ as input lattices, resulting from our basic idea that almost certain values are a bit less true and false. As $\mathcal{L}_1$ and $\mathcal{L}_2$ are totally ordered sets, they are complete lattices and hence infinitely distributive. The resulting bilattice, which we call $\mathcal{FOUR} - \mathcal{C}$, is shown in fig. 3.2. In fig. 3.2 we label nodes of the form $\langle f_x, t_y\rangle$ with $\top_{xy}$,

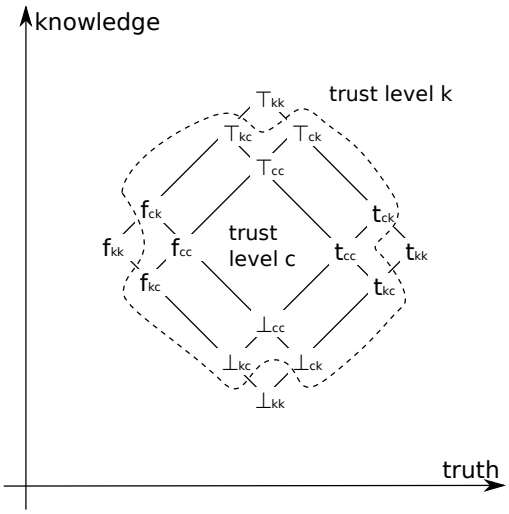


Figure 3.2: $\mathcal{FOUR}-\mathcal{C}$

$\langle t_x, f_y\rangle$ with $\bot_{xy}$, $\langle f_x, f_y\rangle$ with f_{xy} and $\langle t_x, t_y\rangle$ with t_{xy}. On trust level k we have certain, local information. On this level we basically have the usual four truth values of paraconsistent reasoning. on trust level c, we have a second set of four truth values (subscripts cc), which are assigned to cached information. Truth values with mixed subscripts, e.g. kc are inferred, i.e. partially based on both certain and cached information.

3.1.2 Extending OWL2 to logical bilattices

In this section we extend $\mathcal{SROIQ}$, the description logic underlying the proposed OWL2 [GM08], to $\mathcal{SROIQ}-\mathcal{T}$ evaluated on a logical bilattice. The extension towards logical bilattices works analogously to the extension of $\mathcal{SHOIN}$ towards a fuzzy logic as proposed in [Str06]. Operators marked with a dot, e.g. $\dot{\geq}$ are the lattice operators described above, all other operators are the usual (two valued) boolean operators. For two valued operators and a logical bilattice $\mathcal{L}$ we map t to $max_t(\mathcal{L})$ – i.e. the maximal value with respect to the truth order of $\mathcal{L}$ (usually denoted by $\top$) – and f to $min_t(\mathcal{L})$, in order to model that these truth values are absolutely trusted[1]. Please note that while we limit ourselves to $\mathcal{SROIQ}$ here, analogous extensions are possible for $\mathcal{SROIQ}(\mathcal{D})$ to support datatypes. Please also note that we do not include language constructs, which can be expressed by a combination of other constructs defined below. In particular, $Sym(R) = R^- \sqsubseteq R$ and $Tra(R) = R \circ R \sqsubseteq R$.

Definition 9 (Vocabulary) *A vocabulary $V = (N_C, N_P, N_I)$ is a triple where*

- *N_C is a set of OWL classes,*
- *N_P is a set of properties and*
- *N_I is a set of individuals.*

N_C, N_P, N_I need not be disjoint.

A first generalization is that interpretations assign truth values from any given bilattice. In contrast, $\mathcal{SROIQ}$ is defined via set membership of (tuples of) individuals in classes (properties) and uses two truth values only.

Definition 10 (Interpretation) *Given a vocabulary V an* interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \mathcal{L}, \cdot^{\mathcal{I}_C}, \cdot^{\mathcal{I}_P}, \cdot^{\mathcal{I}_i})$ *is a 5-tuple where*

[1] In $\mathcal{FOUR}-\mathcal{T}$ these would be f_∞ and t_∞, but we start with the general case.

- $\Delta^{\mathcal{I}}$ *is a nonempty set called the object domain;*
- $\mathcal{L}$ *is a logical bilattice and* Λ *is the set of truth values in* $\mathcal{L}$
- $\cdot^{\mathcal{I}_C}$ *is the class interpretation function, which assigns to each OWL class* $A \in N_C$ *a function:* $A^{\mathcal{I}_C} : \Delta^{\mathcal{I}} \to \Lambda$*;*
- $\cdot^{\mathcal{I}_P}$ *is the property interpretation function, which assigns to each property* $R \in N_P$ *a function* $R^{\mathcal{I}_P} : \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \to \Lambda$*;*
- $\cdot^{\mathcal{I}_i}$ *is the individual interpretation function, which assigns to each individual* $a \in N_I$ *an element* $a^{\mathcal{I}_i}$ *from* $\Delta^{\mathcal{I}}$*.*

$\mathcal{I}$ *is called a* complete interpretation*, if the domain of every class is* $\Delta^{\mathcal{I}}$ *and the domain of every property is* $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$*.*

The notion of a complete interpretation is needen, because interpretation functions assign a truth value instead of just defining a set membership. In two valued description logics, set membership of an individual in a class corresponds to a truth value of *true* and the default is *false*. In four valued description logics, analogously two sets are used for each class. Hence, by listing only the membership of some individuals, the truth value of class membership is still well defined for all individuals. Here, such a simple convention can not be applied, as we have multiple possible truth values. Instead, truth values must explicitly be assigned for each individual.
We extend the property interpretation function $\cdot^{\mathcal{I}_P}$ to property expressions:

$$(R^-)^{\mathcal{I}_P} \quad = \quad \{(\langle x, y\rangle, u) | (\langle y, x\rangle, u) \in R^{\mathcal{I}_P}\}$$

The second generalization over $\mathcal{SROIQ}$ is the replacement of all quantifiers over set memberships with conjunctions and disjunctions over Λ. We extend the class interpretation function $\cdot^{\mathcal{I}_C}$ to descriptions as shown in table. 3.1.
Satisfaction of axioms in an interpretation $\mathcal{I}$ is defined in table 3.2. With $\circ$ we denote the composition of binary relations. For any function f, $dom(f)$ returns the domain of f. The generalization is analogous to that of $\cdot^{\mathcal{I}_C}$. Note that for equality of individuals, we only need two valued equality.
Satisfiability in $\mathcal{SROIQ} - \mathcal{T}$ is a bit unusual, because when using a logical bilattice we can always come up with interpretations satisfying all axioms by assigning $\top$ and $\bot$. Therefore, we define satisfiability wrt. a truth value:

Definition 11 (Satisfiability) *We say an axiom* E *is* u-satisfiable *in an ontology* O *wrt. a bilattice* $\mathcal{L}$*, if there exists a complete interpretation* $\mathcal{I}$ *of* O *wrt.* $\mathcal{L}$*, which assigns a truth value* $val(E, \mathcal{I})$ *to* E*, such that* $val(E, \mathcal{I}) \geq_k u$*.*
We say an ontology O *is* u*-satisfiable, if there exist a complete interpretation* $\mathcal{I}$*, which* u*-satisfies all axioms in* O *and for each class* C *we have* $|\{a | \langle a, v\rangle \in C \wedge v \geq_t u\}| > 0$*, that means no class is empty.*

Analogously we define consistency wrt. the knowledge order. O is an ontology with axioms collected from multiple sources S_i. Please note that we use a single ontology here to follow the definition of $\mathcal{SROIQ}$, but the composition could happen through imports, or mappings in a network of ontologies. The S_i differ with respect to their trustworthyness. Hence, a partial trust order $\mathcal{T}$ models their trustwortiness. Also note that an analogous is possible for partial intepretations for parts of an ontology. Again we focus on complete interpretations analogous to models in $\mathcal{SROIQ}$.

$$
\begin{array}{rcl}
\top^{\mathcal{I}}(x) & = & t_{yy}, \text{where y is the information source, defining} \top^{\mathcal{I}}(x) \\
\bot^{\mathcal{I}}(x) & = & f_{yy}, \text{where y is the information source, defining} \bot^{\mathcal{I}}(x) \\
(C_1 \sqcap C_2)^{\mathcal{I}}(x) & = & C_1^{\mathcal{I}}(x) \dot{\wedge} C_2^{\mathcal{I}}(x) \\
(C_1 \sqcup C_2)^{\mathcal{I}}(x) & = & C_1^{\mathcal{I}}(x) \dot{\vee} C_2^{\mathcal{I}}(x) \\
(\neg C)^{\mathcal{I}}(x) & = & \dot{\neg} C^{\mathcal{I}}(x) \\
(S^-)^{\mathcal{I}}(x,y) & = & S^{\mathcal{I}}(y,x) \\
(\forall R.C)^{\mathcal{I}}(x) & = & \dot{\bigwedge}_{y \in \Delta^{\mathcal{I}}} R^{\mathcal{I}}(x,y) \dot{\rightarrow} C^{\mathcal{I}}(y) \\
(\exists R.C)^{\mathcal{I}}(x) & = & \dot{\bigvee}_{y \in \Delta^{\mathcal{I}}} R^{\mathcal{I}}(x,y) \dot{\wedge} C^{\mathcal{I}}(y) \\
(\exists R.\mathsf{Self})^{\mathcal{I}}(x) & = & R^{\mathcal{I}}(x,x) \\
(\geq nS)^{\mathcal{I}}(x) & = & \dot{\bigvee}_{\{y_1,\ldots,y_m\} \subseteq \Delta^{\mathcal{I}}, m \geq n} \dot{\bigwedge}_{i=1}^{n} S^{\mathcal{I}}(x,y_i) \\
(\leq nS)^{\mathcal{I}}(x) & = & \dot{\neg} \dot{\bigvee}_{\{y_1,\ldots,y_{n+1}\} \subseteq \Delta^{\mathcal{I}}} \dot{\bigwedge}_{i=1}^{n+1} S^{\mathcal{I}}(x,y_i) \\
\{a_1,\ldots,a_n\}^{\mathcal{I}}(x) & = & \dot{\bigvee}_{i=1}^{n} a_i^{\mathcal{I}} = x
\end{array}
$$

Table 3.1: Extended Class Interpretation Function

$$
\begin{array}{rcl}
(R \sqsubseteq S)^{\mathcal{I}} & = & \dot{\bigwedge}_{x,y \in \Delta^{\mathcal{I}}} R^{\mathcal{I}}(x,y) \dot{\rightarrow} S^{\mathcal{I}}(x,y) \\
(R = S)^{\mathcal{I}} & = & \dot{\bigwedge}_{x,y \in \Delta^{\mathcal{I}}} R^{\mathcal{I}}(x,y) \dot{\leftrightarrow} S^{\mathcal{I}}(x,y) \\
(R_1 \circ \ldots \circ R_n \sqsubseteq S)^{\mathcal{I}} & = & \dot{\bigwedge}_{\langle x_1, x_{n+1} \rangle \in dom(S^{\mathcal{I}})} \dot{\bigvee}_{\{x_2,\ldots,x_n\}} \dot{\bigwedge}_{i=1}^{n} R_i^{\mathcal{I}}(x_i, x_{i+1}) \\
(Asy(R))^{\mathcal{I}} & = & \dot{\bigwedge}_{x,y \in \Delta^{\mathcal{I}}} \dot{\neg}(R^{\mathcal{I}}(x,y) \dot{\wedge} R^{\mathcal{I}}(y,x)) \\
(Ref(R))^{\mathcal{I}} & = & \dot{\bigwedge}_{x \in \Delta^{\mathcal{I}}} R^{\mathcal{I}}(x,x) \\
(Irr(R))^{\mathcal{I}} & = & \dot{\bigwedge}_{x \in \Delta^{\mathcal{I}}} \dot{\neg} R^{\mathcal{I}}(x,x) \\
(Dis(R,S))^{\mathcal{I}} & = & \dot{\bigwedge}_{x,y \in \Delta^{\mathcal{I}}} R^{\mathcal{I}}(x,y) \dot{\rightarrow} \dot{\neg} S^{\mathcal{I}}(x,y) \\
(C \sqsubseteq D)^{\mathcal{I}} & = & \dot{\bigwedge}_{x \in \Delta^{\mathcal{I}}} C^{\mathcal{I}}(x) \dot{\rightarrow} D^{\mathcal{I}}(x) \\
(a : C)^{\mathcal{I}} & = & C^{\mathcal{I}}(a^{\mathcal{I}}) \\
((a,b) : R)^{\mathcal{I}} & = & R^{\mathcal{I}}(a^{\mathcal{I}}, b^{\mathcal{I}}) \\
a \approx b & = & a^{\mathcal{I}} = b^{\mathcal{I}} \\
a \not\approx b & = & a^{\mathcal{I}} \neq b^{\mathcal{I}}.
\end{array}
$$

Table 3.2: Satisfaction of Axioms

Definition 12 (Consistency) *Let $\mathcal{I}$ be a complete interpretation, O an ontology, which is composed from multiple data sources $\{S_1, ..., S_n\}$ and $\mathcal{T}$ a trust order over $\{S_1, ..., S_n\}$. Let $source(E)$ denote the $\mathcal{T}$-maximal datasource, from which axiom E originates.*
*We say O is u-consistent, if there exists an $\mathcal{I}$, which assigns a truth value $val(E, \mathcal{I})$ to all axioms E in O, such that $u \leq_k val(E, \mathcal{I})$. $\mathcal{I}$ is called a u-*model *of O.*
We say O is consistent, if there exist an $\mathcal{I}$, which assigns a truth value $val(E, \mathcal{I})$ to all axioms E in O, such that $\forall x : val(E, \mathcal{I}) \notin \{\top_x, \bot_x\}$. We say $\mathcal{I}$ is a model of O.

Finally, we define entailment:

Definition 13 (Entailment) *O entails a $\mathcal{SROIQ} - \mathcal{T}$ ontology O' ($O \models O'$), if every model of O is also a model of O'. O and O' are equivalent if O entails O' and O' entails O.*

The following theorem shows that we have indeed defined a strict extension of $\mathcal{SROIQ}$. Please not that the following theorem only holds for $\mathcal{SROIQ}$ as logical bilattice and for ontologies, which are consistent in standard two valued logics.

Theorem 3 *If $\mathcal{FOUR}$ is used as logical bilattice, $\mathcal{SROIQ} - \mathcal{T}$ is isomorphic to $\mathcal{SROIQ}$.*

Proof 3 (sketch) *From a model of a $\mathcal{SROIQ} - \mathcal{T}$ ontology O wrt. $\mathcal{FOUR}$, we can derive a model for the same ontology in $\mathcal{SROIQ}$ by doing the following steps:*

- *For each class C, replace $C(a) = t$ by $a \in C$ and $C(a) = f$ by $a \notin C$. Analogous for properties.*
- *As O is consistent, we do not have $\top$ and $\bot$ truth values in a model.*
- *The only connectives used in the ontology language are $\neg, \vee, \wedge$, these are equivalent to their boolean counterparts in $\mathcal{FOUR}$.*
- *replace "trust-consistent" in $\mathcal{SROIQ} - \mathcal{T}$ with "consistent" in $\mathcal{SROIQ}$. Analogous for satisfiability.*

For a complete proof we need to show for every rule in tables 3.1 and 3.2 that we can transform it to the $\mathcal{SROIQ}$ form (wrt. $\mathcal{FOUR}$) by replacing conjunctions and disjunctions by quantifiers over set membership or inclusion.

3.2 Applications

In addition to inconsistencies, which will almost certainly occur in the Semantic Web, be will also be faced with information sources, which are of varying trustworthiness. Hence, we define a logical bilattice, which allows for reasoning with trust orders. In the previous chapter we have focused on two levels of reliability of information. More generally, we would like to be able to infer multiple levels of trust in a distributed setting.

Definition 14 (Trust Order) *A trust order $\mathcal{T}$ is a partial order over a finite set of information sources with a maximal element, called ∞.*

∞ is the information source with the highest trust level, assigned to local data. For any two information sources a and b comparable wrt. $\mathcal{T}$, we have a $\mathcal{FOUR} - \mathcal{C}$ lattice as described above, with the less trusted information source corresponding to the inner part of the lattice. Extended to multiple information sources, this results in a situation as depicted in fig. 3.3, where the outermost bilattice corresponds to local, fully trusted information.
If a and b are not comparable we introduce virtual information sources inf_{ab} and sup_{ab}, such that

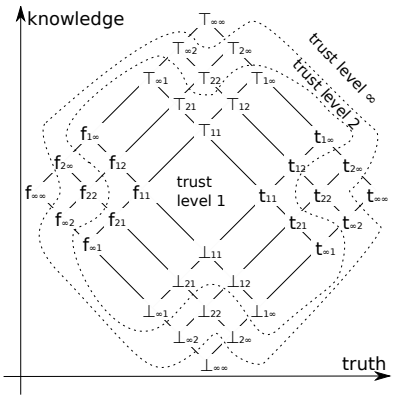


Figure 3.3: $\mathcal{FOUR}-\mathcal{T}$

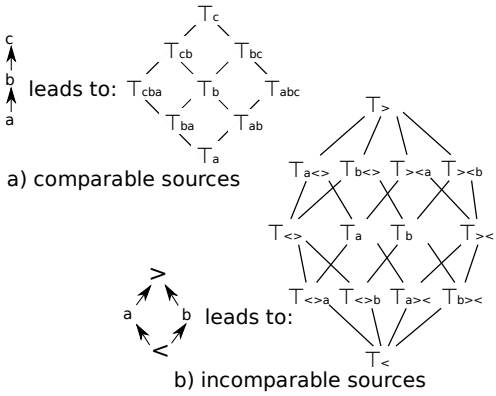


Figure 3.4: Virtual Sources

- $inf_{ab} < a < sup_{ab}$ and $inf_{ab} < b < sup_{ab}$;
- $\forall_{c<a,c<b} : c < inf_{ab}$ and
- $\forall_{d>a,d>b} : d > sup_{ab}$

To understand the importance of this last step, assume that $c > a > d$ and $c > b > d$ and a, b are incomparable. Then the truth value of $a \vee b$ would have a trust level of c, as c is the supremum in the trust order. Obviously this escaping to a higher trust level is not desirable. Instead, the virtual information sources represent that we need to trust both, a and b, if we believe in the computed truth value. We illustrate this situation in fig. 3.4 (We abbreviate inf_{ab} by $<$ and sup_{ab} by $>$).

In the general case (≥ 3 incomparable sources) such a trust order results in a non-distributive lattice. This can be fixed, however, by introducing additional virtual nodes. The basic idea here is to create a virtual node for each element in the powerset of the incomparable sources, with set inclusion as the order. We will call this modified trust order *completed*. We can again derive a complete lattice from the completed trust order. As it can become quite large, we only show for $\top$ how a fragment of the logical bilattice is derived from the trust order for the case of two incomparable information sources in fig. 3.4.

Using the same method as in the previous chapter, we can construct the corresponding bilattice from a given completed truth order as follows: Given a trust order $\mathcal{T}$, generate a lattice $\mathcal{L}_1$, such that

- $f_a <_{\mathcal{L}_1} f_b$, iff $a >_{\mathcal{T}} b$;
- $t_b <_{\mathcal{L}_1} t_a$, iff $a >_{\mathcal{T}} b$ and
- $\forall a : f_a <_{\mathcal{L}_1} t_a$.

The result is a lattice with t_∞ and f_∞ as maximal and minimal elements. Now create the logical bilattice $\mathcal{L}$ from $\mathcal{L}_1 \times \mathcal{L}_1$ as described in the previous section.

in Figure 3.3 we see a bilattice resulting from a strict trust ordering, i.e. every pair of information sources in the trust order is comparable. In general, it will not be possible to come up with such a strict ordering. For example, interpretations of news may vary also among very trustworthy newspapers, depending on their political background. Hence, we might have a trust ordering with incomparable information sources. This case is demonstrated in Figure 3.4. Case a) shows the $\top$ part of the logical bilattice from Figure 3.3. Case b) shows the same part for a bilattice resulting from a trust ordering, in which sources a and b are incomparable. Their supremum and infimum are denoted by $<$ and $>$ respectively. As we can see, trust-bilattices can grow quite complex. As users of reasoning services will be interested in trust levels only, they need not be faced with all these values, however. Further, the bilattice needs not be materialized for reasoning, as many truth values will not even be used. However, to be complete, all these values must be defined.

3.2.1 Resolving Inconsistencies

Inconsistencies in ontologies often emerge, when ontologies are integrated from various sources using ontology modules, ontology mappings and similar mechanisms [P. 05]. In this section we propose to use trust based reasoning for inconsistency resolution. Trust levels will be used to select axioms to remove for repairing the ontology. The actual ontology repair has been described in NeOn deliverable [QHJ08].

Definition 15 (Maximally Trust Consistent Interpretation) *We say an interpretation $\mathcal{I}$ is* maximally trust consistent*, if it does not assign any artificial $\top$ values, i.e. $\top_{xy}$ with $x \neq y$. An ontology O is said to be maximally trust consistent, if it has an interpretation $\mathcal{I}$, which is maximally trust consistent.*

This means, a trust maximal interpretation can still be inconsistent, but such inconsistency then arises from information obtained from a single information source. We now define, how a maximally trust consistent ontology can be derived from any given ontology.

Various approaches for *repairing* inconsistent ontologies have been proposed. In most approaches, axioms are removed from the ontology until the rest is a consistent ontology (cf.[P. 05]). There usually are multiple possible choices for axioms to remove. While this might not seem too bad in our example, consider a similar scenario involving a red traffic light and we accidentally remove $RedLightSituation \sqsubseteq BetterBreakSituation$. Here trust based reasoning comes into play. We use the trust level as input to a selection function, which determines axioms to be removed, a point left open in [P. 05].

Definition 16 (Minimal Inconsistent Subontology) *Let O be an inconsistent ontology. A* minimal inconsistent subontology $O' \subseteq O$ *is an inconsistent ontology, such that every $O'' \subset O'$ is consistent.*

Now trust maximal consistency is reestablished by iteratively removing all axioms with the lowest trust level from O', until the resulting ontology is trust maximal consistent. This captures the idea, that in the case of an inconsistency, humans tend to ignore lowly trusted information first.

If a trust maximal consistent ontology still is inconsistent, we need a different selection function. However, in this case already the *local* knowledge is inconsistent. A similar situation arises, if we have an inconsistency resulting from two incomparable information source. In the latter case, however, we can choose to discard information from both. Of cause, depending on the actual application, we can also use a more sophisticated selection function, choosing among axioms on the lowest trust level.

Chapter 4

A Forgetting-based Approach for Reasoning with Distributed Ontologies

4.1 Distributed systems

In this section, we introduce the notion of a distributed system defined in [ES07]. Note that our definitions are compatible with the mapping metamodel given in NeOn deliverable D1.1.2 [HBP+07].

Given two ontologies O_1 and O_2, describing the same or largely overlapping domains of interest. We can define the correspondences between their elements.

Definition 17 *Let O_1 and O_2 be two ontologies, Q be a function that defines sets of mappable elements $Q(O_1)$ and $Q(O_2)$. A correspondence is a 4-tuple $\langle e, e', r, \alpha \rangle$ such that $e \in Q(O_1)$ and $e' \in Q(O_2)$, r is a semantic relation, and α is a confidence value from a suitable structure $\langle D, \leq \rangle$, such as a lattice.*

In Definition 17, there is no restriction on function Q, semantic relation r and domain D. However, in our work, we only consider correspondences between concepts and restrict r to be one of the semantic relations from the set $\{\equiv, \sqsubseteq, \sqsupseteq\}$, and assume $D = [0.0, 1.0]$.

From a set of correspondences, we can define the notion of a mapping as follows. We follow the definition of a mapping given in [HBP+07].

Definition 18 *Given ontologies O_1 and O_2, let Q be a function that is given in Definition 17. $\mathcal{M}$ is a set of correspondences. Then $\mathcal{M}$ is a mapping between O_1 and O_2 iff for all correspondences $\langle e, e', r, \alpha \rangle \in \mathcal{M}$ we have $e \in Q(O_1)$ and $e' \in Q(O_2)$.*

That is, a mapping is a set of correspondences whose elements are matchable.

Given a mapping between two ontologies O_1 and O_2, we can define the notion of a distributed system[1] [ZE06].

Definition 19 *A distributed system is a triple $\mathcal{D} = \langle O_1, O_2, \mathcal{M} \rangle$, where O_1 and O_2 are ontologies and $\mathcal{M}$ is a mapping between them. We call O_1 the source ontology and O_2 the target ontology.*

Given a distributed system, there is no unique semantics for it. Three semantics for distributed systems have been proposed in [ZE06]. The first one, called simple distributed semantics, considers the whole distributed system as a coherent knowledge base which can be interpreted in a single domain. The second one is called an integrated distributed semantics, where each local knowledge representation is interpreted in its own domain but these interpretation are then correlated in a global domain. The third one is called a contextualized

[1] In [ZE06], they do not restrict to two ontologies. To simplify discussions, we consider only two ontologies in our work.

distributed semantics, where there is no global domain of interpretation: each local ontologies imports knowledge from other ontologies in its own context. They have developed an algorithm for consistency checking in the logic based on the integrated distributed semantics in [ZD08]. Another important semantics for a distributed system is defined by Distributed Description Logics (DDL) [BS03]. Algorithms for reasoning tasks in DDL, such as consistency checking, have been implemented as an extension DRAGO system [ST05]. A global semantics for a distributed system is given in [MS07]. This semantics is the same as that of the mapping system described in [HM05], which has been discussed in NeOn deliverable D1.1.2 [HBP+07].

4.2 Forgetting

4.2.1 Variable forgetting

Let L be a propositional language constructed from a finite alphabet P of propositional symbols, the Boolean constants $\top$ (true) and $\bot$ (false), using the usual operators $\neg$ (not), $\vee$ (or) and $\wedge$ (and). An interpretation is a total function from P to $\{true, false\}$. The classical consequence relation is denoted by $\vdash$. Given a propositional formula ϕ, $Var(\phi)$ denotes the set of propositional variables occuring in ϕ. $\phi_{x\leftarrow 0}$ (resp. $\phi_{x\leftarrow 1}$) denotes the formula obtained by replacing in ϕ every occurence of variable x by $\bot$ (resp. $\top$).

Variable forgetting, which is also known as projection, is defined in [LR94] and is applied to resolve inconsistency in [LM02].

Definition 20 *Let ϕ be a formula over P and $V \subseteq P$ be a set of propositional symbols. The forgetting of V in ϕ, denoted as* $\mathsf{Forget}(V, \phi)$*, is a formula over P, defined inductively as follows:*

- $\mathsf{Forget}(\varnothing, \phi) \equiv \phi$;
- $\mathsf{Forget}(\{x\}, \phi) \equiv \phi_{x\leftarrow 0} \vee \phi_{x\leftarrow 1}$;
- $\mathsf{Forget}(\{x\} \cup V, \phi) \equiv \mathsf{Forget}(V, \mathsf{Forget}(\{x\}, \phi))$.

$\mathsf{Forget}(V, \phi)$ is the logically strongest consequence of ϕ which is independent of V. By forgetting a set of variables in a formula ϕ we get a formula which is inferentially weaker than ϕ, i.e. $\phi \vdash \mathsf{Forget}(V, \phi)$. For example, $\mathsf{Forget}(\{a\}, \neg a \wedge b) \equiv b$ and $\mathsf{Forget}(\{a\}, a \vee b) \equiv \top$.

A semantic definition of forgetting is given in [LR94]. Given two interpretations ω and ω', suppose x is a variable, we define $\omega \sim_x \omega'$ iff ω and ω' agree on everything except possibly on the truth value of x. More formally, $\omega \sim_x \omega'$ iff for any variable y distinct from x, $\omega(y) = \omega'(y)$.

Definition 21 *Let ϕ be a formula over P and x be a variable in P. A formula ψ is the result of forgetting of x in ϕ iff for any interpretation ω, ω is a model of ψ iff there is a model ω' of ϕ such that $\omega \sim_x \omega'$.*

We denote by $\mathsf{Forget}_S(x, \phi)$ the result of forgetting x in ϕ using the forgetting defined in Definition 21. It has been shown in [LR94] that $\mathsf{Forget}(x, \phi) = \mathsf{Forget}_S(x, \phi)$, for any x and ϕ. That is, the syntactical and semantic definitions of forgetting coincide. Given a set $V = \{x_1, ..., x_n\}$ of variables in P, the result of forgetting of V in ϕ, written $\mathsf{Forget}(V, \phi)$, is defined inductively as $\mathsf{Forget}(x_n, \mathsf{Forget}(\{x_1, ..., x_{n-1}\}, \phi)$. It has been shown in [LR94] that the order of variables does not matter.

4.2.2 Forgetting in description logics

In this section, we define forgetting in a TBox of a DL knowledge base. The notion of forgetting can be applied to any description logic. Here, a concept name plays the same role as a variable in propositional logic. In particular, we define the forgetting of a set of concept names in a TBox. We first consider a semantic definition of forgetting in DLs given in [WWTP08]. Since computation of the result of semantic forgetting is very difficult, we then give a definition of syntactical forgetting, which is easy to compute.

Semantic forgetting

We first introduce the relation $\sim_A$ and the notion of concept forgetting given in [WWTP08]: let A be a concept name in a DL language $\mathcal{L}$, and $\mathcal{I}$ and $\mathcal{I}'$ interpretations of $\mathcal{L}$. We define $\mathcal{I} \sim_A \mathcal{I}'$ iff $\mathcal{I}$ and $\mathcal{I}'$ agree on all concept names and role names except possibly on A. The result of forgetting about A in $\mathcal{T}$, denoted as $\texttt{forget}(\mathcal{T}, A)$, is defined in a model-theoretical way as follows: $\texttt{forget}(\mathcal{T}, A)$ is a TBox on the signature $\texttt{Sig}(\mathcal{T}) \setminus \{A\}$ and any interpretation $\mathcal{I}'$ is a model of $\texttt{forget}(\mathcal{T}, A)$ iff there is a model $\mathcal{I}$ of $\mathcal{T}$ such that $\mathcal{I} \sim_A \mathcal{I}'$. It has been shown in [WWTP08] that when we forget a set $\mathbf{A} = \{A_1, ..., A_n\}$ of concept names, the order of concept forgetting will not influence the final result of forgetting. Therefore, we can define $\texttt{forget}(\mathcal{T}, \mathbf{A}) = \texttt{forget}(...(\texttt{forget}(\mathcal{T}, A_1), ...), A_n)$. In [WWTP08], an algorithm is given to compute the result of concept forgetting in *DL-Lite*, a family of DLs that provide tractable reasoning. However, for more expressive DLs, the result of concept forgetting may not be expressed in the same language [KLWW08]. Another restriction of the concept forgetting defined in [WWTP08] is that the result of forgetting of a concept in a DL-Lite ontology with non-empty ABox may not be expressible in the same language. Next, we propose a definition of syntactical forgetting, which can be applied to any DL and is easy to compute.

Syntactical forgetting

We use $\mathsf{Con}(C)$ to denote the concept names in a concept C. $C_{A\leftarrow\top}$ (resp. $C_{A\leftarrow\bot}$) denotes the concept obtained by replacing in C every occurrence of concept name A by $\top$ (resp. $\bot$).

A straightforward way to forget a concept name A in a concept C can be given as follows: $\mathsf{Forget}(\{A\}, C)$=$C_{A\leftarrow\top} \sqcup C_{A\leftarrow\bot}$. However, this definition suffers from a problem. That is, by forgetting a concept name A in a concept C in a TBox $\mathcal{T}$, we do not necessarily obtain a concept which is logically weaker than C, i.e. $\mathcal{T} \models C \sqsubseteq \mathsf{Forget}(\{A\}, C)$, a very important property of forgetting. The main reason for this problem is that the DL concept constructors include value restrictions on role names and existential restriction on role names. Let us look at an example. Suppose $C = \forall R.(A \sqcap \forall R'.\neg A)$ and we want to forget A. Since $C_{A\leftarrow\top} = \forall R.(\top \sqcap \forall R'.\bot) = \forall R.(\forall R'.\bot)$ and $C_{A\leftarrow\bot} = \forall R.(\bot \sqcap \forall R'.\top) = \forall R.\bot$, we have $\mathsf{Forget}(\{A\}, C) = (\forall R.(\forall R'.\bot)) \sqcup \forall R.\bot$. In this case, we do not necessarily have $\mathcal{T} \models C \sqsubseteq \mathsf{Forget}(\{A\}, C)$.

We consider the following definition of forgetting.

Definition 22 *Let C be a (complex) concept and A a concept name. Suppose C is in negation normal form, i.e. negation can only appear in the front of a concept name. The forgetting of A in C, denoted $\mathsf{Forget}(\{A\}, C)$, is a concept obtained by replacing every occurrence of A and $\neg A$ by $\top$.*

In Definition 22, it is very important that the concept C is transformed into its negation normal form before we apply the forgetting operator to it. Otherwise, we cannot ensure that $C \sqsubseteq \mathsf{Forget}(\{A\}, C)$. For example, suppose $C = \neg(A_1 \sqcap B_1)$ and we want to forget A_1. If we replace A_1 by $\top$ without transforming C into its negation normal form, then $C' = \neg B_1$, which is subsumed by C.

We give a running example to illustrate the notion of forgetting.

Example 2 *Let us consider a TBox*
$\mathcal{T} = \{\mathsf{Professor} \sqcup \mathsf{PostDoctor} \sqcup \mathsf{ResearchAssistant} \sqsubseteq \mathsf{Employee}$, $\mathsf{PhDStudent} \sqsubseteq \mathsf{ResearchAssistant}$, $\mathsf{PhDSupervisor} \equiv \mathsf{Professor} \sqcap \exists \mathsf{isSupervisorof}.\mathsf{PhDStudent}\}$. *Then,*
$\mathsf{Forget}(\{\mathsf{Professor}\}, \mathsf{Professor} \sqcup \mathsf{PostDoctor} \sqcup \mathsf{ResearchAssistant}) = \top$ *and*
$\mathsf{Forget}(\{\mathsf{Professor}\}, \mathsf{Professor} \sqcap \exists \mathsf{isSupervisorof}.\mathsf{PhDStudent}) = \exists \mathsf{isSupervisorof}.\ \mathsf{PhDStudent}$.

The following proposition tells us that the forgetting of A in C indeed results in a concept which is a super concept of C.

Proposition 7 *Let C be a concept in a TBox $\mathcal{T}$ and A be a concept name. Then $\mathcal{T} \models C \sqsubseteq \mathsf{Forget}(\{A\}, C)$.*

We now define the forgetting of a set of concept names in a concept.

Definition 23 *Let C be a concept, and let S_N be a finite set of concept names. The forgetting of S_N in C, denoted* $\mathsf{Forget}(S_N, C)$*, is defined inductively as follows:*

- $\mathsf{Forget}(\varnothing, C)$=$C$*;*
- $\mathsf{Forget}(\{A\} \cup S_N, C) = \mathsf{Forget}(S_N, \mathsf{Forget}(\{A\}, C))$.

Example 3 *(Example 2 Continued) Let* $C = \mathsf{Professor} \sqcap \exists \mathsf{isSupervisorof}.\mathsf{PhDStudent}$*. Suppose we want to forget* Professor *and* PhDStudent *in C, i.e., $S_N = \{\mathsf{Professor}, \mathsf{PhDStudent}\}$, then we have* $\mathsf{Forget}(S_N, C) = \exists \mathsf{isSupervisorof}.\top$*.*

Unlike the forgetting notion in propositional theories, we do not have a corresponding model theoretic semantics for concept forgetting. However, we have the following properties which precisely characterize our notion of concept forgetting.

Proposition 8 *Let C, D and E be concepts, and let A and A' be two concept names. We then have*

1. *If $C = A$ or $\neg A$, then* $\mathsf{Forget}(\{A\}, C) = \top$*;*
2. *If $C = D \sqcap E$ (or $D \sqcup E$), where $A \in \mathsf{Con}(D)$ and $A \notin \mathsf{Con}(E)$, then* $\mathsf{Forget}(\{A\}, C) = \mathsf{Forget}(\{A\}, D) \sqcap E$ *(or* $\mathsf{Forget}(\{A\}, C) = \mathsf{Forget}(\{A\}, D) \sqcup E$*).*
3. *If $C = A \sqcup E$, then* $\mathsf{Forget}(\{A\}, C) = \top$*.*

The proof of Proposition 8 is clear by the definition of forgetting. Item 1 says that if C is a concept name or full negation of a concept name, then by forgetting it we get a top concept. Item 2 shows that forgetting a concept name A is a concept which is a conjunction (or disjunction) will not influence the conjunct (or disjunct) which is irrelevant to the concept name. Item 3 is a disjunction of the form $A \sqcup E$, then by forgetting A we get a top concept.

We now consider the weakening of a TBox $\mathcal{T}$ by forgetting a concept name A. A first thought would be that we simply forget A in every concept appearing in $\mathcal{T}$. However, this approach may cause some problems. Suppose a terminology axiom $C \sqsubseteq D$ is in $\mathcal{T}$ such that $C = A \sqcap B$, and $A \notin \mathsf{Con}(D)$ and $A \notin \mathsf{Con}(B)$. By forgetting A in C we get a new axiom $B \sqsubseteq D$. It is easy to check that $B \sqsubseteq D$ infers $C \sqsubseteq D$ whilst the converse does not always hold. Therefore, $C \sqsubseteq D$ is not weakened. On the contrary, it is reinforced. To solve this problem, we can equivalently transform every axiom of the form $C \sqsubseteq D$ into $\top \sqsubseteq \neg C \sqcup D$. According to Proposition 7, we have the following corollary.

Corollary 1 *Let ϕ be a terminology axiom of the form $\top \sqsubseteq C$ and A a concept name. Suppose $A \in \mathsf{Con}(C)$, then $\top \sqsubseteq C \models \top \sqsubseteq \mathsf{Forget}(\{A\}, C)$, but not vice versa.*

Corollary 1 tells us that an axiom is weakened by forgetting a concept name in the right-side concept.

We give the notion of forgetting in a TBox.

Definition 24 *Let $\mathcal{T}$ be a TBox. Let ϕ be a terminology axiom in $\mathcal{T}$ which has been transformed into the form $\top \sqsubseteq C$ and S_N a finite set of concept names. The forgetting of S_N in ϕ is defined as* $\mathsf{Forget}(S_N, \phi) = \top \sqsubseteq \mathsf{Forget}(S_N, C)$ *and the forgetting of S_N in $\mathcal{T}$ is defined as* $\mathsf{Forget}(S_N, \mathcal{T}) = \{\mathsf{Forget}(S_N, \phi) : \phi \in \mathcal{T}, Sig(\phi) \cap S_N \neq \varnothing\}$*, where $Sig(\phi)$ denotes the signature of ϕ.*

By Corollary 1, it is clear that we have that every axiom in $\mathsf{Forget}(S_N, \mathcal{T})$ can be inferred from $\mathcal{T}$, for any S_N.

Example 4 *(Example 2 Continued) Suppose we want to forget* Professor *in* $\mathcal{T}$. *We need some pre-processing. First,* Professor⊔PostDoctor⊔ResearchAssistant⊑ Employee *is transformed into* $\top \sqsubseteq$ (¬Professor ⊓ ¬PostDoctor ⊓ ¬ResearchAssistant) ⊔ Employee. *Second, we split* PhDSupervisor ≡ Professor⊓∃isSupervisorof.PhDStudent *into*
PhDSupervisor ⊑ Professor⊓∃isSupervisorof.PhDStudent *and*
Professor⊓∃isSupervisorof.PhDStudent ⊑ PhDSupervisor, *then transform them into* $\top \sqsubseteq$ ¬PhDSupervisor ⊔ (Professor ⊓ ∃isSupervisorof.PhDStudent) *and* $\top \sqsubseteq$ ¬Professor ⊔ ¬∃isSupervisorof.PhDStudent ⊔ PhDSupervisor. *Therefore, we have*

$$\begin{aligned}\text{Forget}(\{\text{Professor}\}, \mathcal{T}) \;=\; &\{\top \sqsubseteq (\neg\text{PostDoctor} \sqcap \neg\text{ResearchAssistant}) \sqcup \text{Employee},\\ &\top \sqsubseteq \neg\text{PhDSupervisor} \sqcup \exists\text{isSupervisorof.PhDStudent}\\ &\text{PhDStudent} \sqsubseteq \text{ResearchAssistant}\}.\end{aligned}$$

4.3 Recoveries for Distributed Systems

In this section, we apply the forgetting-based approach for resolving inconsistency in a propositional knowledge base [LM02] to deal with inconsistency in a distributed system by forgetting concepts in TBoxes of local ontologies. We do not consider inconsistency which is caused only by inter-ABoxes as this kind of inconsistency may be better addressed by performing some kind of weakening on the ABoxes of local ontologies. Dealing with this problem is also important, but it is out of scope for this work.

4.3.1 Recoveries

In [LM02], two key notions of the approach are forgetting context and recovery vector. The former one is a collection of sets of variables to be forgotten in each formula from the knowledge base, and the latter one is a vector of subsets of the set of all the variables appearing in the knowledge base, which satisfies the constraints in the forgetting context. Forgetting of the variables in the recovery vector will result in a consistent knowledge base. We adapt the definitions of forgetting context and recovery vector for a distributed system.

Definition 25 *Given a distributed system* $\mathcal{D} = \langle O_1, O_2, \mathcal{M}\rangle$, *where* $O_1 = \langle \mathcal{T}_1, \mathcal{R}_1, \mathcal{A}_1\rangle$ *and* $O_2 = \langle \mathcal{T}_2, \mathcal{R}_2, \mathcal{A}_2\rangle$ *are ontologies and* $\mathcal{M}$ *is a mapping between them, a forgetting context* $\mathcal{C}_\mathcal{D}$ *for it is a triple* $\langle F, G, H\rangle$ *where:*

- $F = \langle F_1, F_2\rangle$ *such that* $F_i \subseteq N_C$ *with* $i = 1, 2$, *and for each* i, F_i *is the set of concept names that can be forgotten in* $\mathcal{T}_i$.
- $G = \langle G_1, G_2\rangle$ *such that for each* i, $G_i = \{(A_i, A_i^{'}) : A_i, A_i' \in CN(\mathcal{T}_i)\}$, *where* $CN(\mathcal{T}_i)$ *is the set of all concept names in* $\mathcal{T}_i$ *and a pair* (A_i, A_i') *in* G_i *means that* A_i' *is meaningful only if* A_i *exists;*
- $H \subseteq N_C \times \{1, 2\} \times N_C \times \{1, 2\}$ *is a set of quadruples* (A, i, B, j) *such that* $A \in F_i$, $B \in F_j$ *and* $i \neq j$, *which means that if* A *is forgotten in* F_i, *then* B *must be forgotten as well in* F_j.

F_i imposes the constraints over the concept names which can be forgotten in $\mathcal{T}_i$. That is, if a concept name $A \notin F_i$, then forgetting A in $\mathcal{T}_i$ is forbidden. G_i contains the pairs of concept names in local TBox $\mathcal{T}_i$ such that the existence of the former is meaningful or significant only in presence of the latter. In other words, forgetting the latter imposes to forget the former. H imposes the constraints over inter-TBoxes. It forces that if A is forgotten in $\mathcal{T}_i$ then B should also be forgotten in $\mathcal{T}_j$. The forgetting context is dependent on the application at hand. A simple example for forgetting context is a *standard* forgetting context which is given by $F_i = N_C$ for every $i = 1, ..., n$, $G_i = \varnothing$ for every $i = 1, ..., n$, and $H = \varnothing$. Note that our approach does currently not include weakening of bridge rules. Such a weakening may be preferable in some situations (for example, to

improve mappings created by different matching systems [CM07]), but its treatment is out of scope for this paper.

We next define a recovery vector for a distributed system. It is a vector of subsets of concept names of N_C, which satisfies the constraints in the forgetting context. In this definition, we need to specify the semantics of a distributed system.

Definition 26 *Given a distributed system $\mathcal{D} = \langle O_1, O_2, \mathcal{M}\rangle$ with Cn as its consequence relation, and a forgetting context $\mathcal{C}_\mathcal{D}$, a recovery vector (or recovery for short) $\overrightarrow{V}$ for $\mathcal{D}$ given $\mathcal{C}_\mathcal{D}$ is a vector $\overrightarrow{V} = \langle V_1, V_2\rangle$ of subsets of N_C s.t.:*

- *for every $i = 1, 2$, $V_i \subseteq \mathsf{Con}(\mathcal{T}_i)$,*
- *for every $i = 1, 2$, for any $(A, A') \in G_i$, if $A \in V_i$ then $A' \in V_i$,*
- *for every $(A, i, B, j) \in H$, $A \in V_i$ implies $B \in V_j$,*
- *$\mathcal{D}|\overrightarrow{V} = \langle\{\langle \mathsf{Forget}(V_i, \mathcal{T}_i), \mathcal{R}_i, \mathcal{A}_i\rangle\}_{i=1,2}, \mathfrak{B}\rangle$ is consistent.*

$\mathcal{D}|\overrightarrow{V}$ is called the cure of $\overrightarrow{V}$ for $\mathcal{D}$ given $\mathcal{C}_\mathcal{D}$. We denote the set of all recoveries for $\mathcal{D}$ given $\mathcal{C}_\mathcal{D}$ as $\mathcal{R}_{\mathcal{C}_\mathcal{D}}(\mathcal{D})$.

According to the definition of recovery, the cure of $\overrightarrow{V}$ for $\mathcal{D}$ is always consistent if the recovery $\overrightarrow{V}$ exists. When $\mathcal{R}_{\mathcal{C}_\mathcal{D}}(\mathcal{D}) = \varnothing$, we say that $\mathcal{D}$ is not recoverable $w.r.t.$ $\mathcal{C}_\mathcal{D}$. This may happen when some concepts, which cannot be forgotten according to the forgetting context, must be forgotten to restore consistency.

We give an example to illustrate the definitions in this section by considering DDL as the semantics of a distributed system. We assume that the reader is familiar with DDL syntax and semantics which can be found in paper [BS03].

Example 5 *Suppose we have a distributed knowledge base $\mathcal{D} = \langle\{O_1, O_2\}, \{\mathfrak{B}_{12}, \mathfrak{B}_{21}\}\rangle$, where*
(1) $O_1 = \langle\mathcal{T}_1, \mathcal{R}_1, \mathcal{A}_1\rangle$:

- $\mathcal{R}_1 = \varnothing$*;*
- $\mathcal{T}_1 = \{\mathsf{Professor} \sqcup \mathsf{PostDoctor} \sqcup \mathsf{ResearchAssistant} \sqsubseteq \mathsf{Employee}$, $\mathsf{PhDStudent} \sqsubseteq \mathsf{ResearchAssistant}$, $\mathsf{PhDSupervisor} \equiv \mathsf{Professor} \sqcap \exists \mathsf{isSupervisorof}.\mathsf{PhDStudent}\}$*;*
- $\mathcal{A}_1 = \{\mathsf{PhDStudent}(Mary), \mathsf{PhDSupervisor}(Tom)\}$*;*

(2) $O_2 = \langle\mathcal{T}_2, \mathcal{R}_2, \mathcal{A}_2\rangle$:

- $\mathcal{R}_2 = \varnothing$*;*
- $\mathcal{T}_2 = \{\mathsf{Professor} \sqcup \mathsf{Lecturer} \sqsubseteq \mathsf{AcademicStaff}, \mathsf{Student} \sqcap \mathsf{AcademicStaff} \sqsubseteq \bot, \mathsf{PhDStudent} \sqsubseteq \mathsf{Student}$, $\mathsf{PhDSupervisor} \equiv \mathsf{AcademicStaff} \sqcap \exists \mathsf{isSupervisorof}.\mathsf{PhDStudent}\}$*;*
- $\mathcal{A}_2 = \{\mathsf{PhDStudent}(John)\}$*;*

(3) $\mathfrak{B}_{21} = \{2 : \mathsf{AcademicStaff} \xrightarrow{\sqsupseteq} 1 : \mathsf{Employee}, 2 : \mathsf{PhDStudent} \xrightarrow{\sqsupseteq} 1 : \mathsf{PhDStudent}, 2 : \neg\mathsf{AcademicStaff} \xrightarrow{\sqsubseteq} 1 : \neg\mathsf{Employee}, 2 : \neg\mathsf{PhDStudent} \xrightarrow{\sqsubseteq} 1 : \neg\mathsf{PhDStudent}\}$ and $\mathfrak{B}_{12} = \varnothing$. O_1 and O_2 are two University ontologies which are connected by bridge rules.
Let $\mathcal{D}' = \langle\{\mathcal{T}_1, \mathcal{T}_2\}, \{\mathfrak{B}_{12}, \mathfrak{B}_{21}\}\rangle$. Since $\mathcal{D}' \models_d \mathsf{PhDStudent} \sqsubseteq \bot$ and $\mathsf{PhDStudent}(Mary) \in \mathcal{A}_1$, $\mathcal{D}$ is inconsistent. We have the following forgetting context $\mathcal{C}_\mathcal{D}$ for $\mathcal{D}$:

- *$F = \langle F_i\rangle_{i=1,2}$, where $F_i =$*
  $\{\mathsf{AcademicStaff}, \mathsf{Employee}, \mathsf{Student}, \mathsf{Professor}, \mathsf{Lecturer},$
  $\mathsf{ResearchAssistant}, \mathsf{PhDStudent}, \mathsf{PhDSupervisor}\}_{i=1,2}$.

- G=$\langle G_i \rangle_{i=1,2}$, *where*
  G_1={(PhDStudent, PhDSupervisor)}; G_2={(PhDStudent, PhDSupervisor)}.
- $H = \{(\text{PhDStudent}, 1, \text{PhDStudent}, 2)\}$.

For each local ontology, all the concept names in it can be forgotten. G_1 *is used to show that the concept* PhD supervisor *is not meaningful if the concept* PhD student *or* professor *is forgotten.* G_2 *can be similarly explained.* H *ensures that the concepts* PhD student, professor, student *in* O_1 *are respectively equal to* PhD student, professor, student *in* O_2 $w.r.t.$ *weakening from the point of view of* O_2.
Given the forgetting context $\mathcal{C}_\mathcal{D}$, *the following are two recoveries for* $\mathcal{D}$:

- $\overrightarrow{V}_1 = \langle V_1, V_2 \rangle$, *where*
  $V_1 = \{\text{ResearchAssistant}\}$ *and* $V_2 = \varnothing$:
  $\text{Forget}(V_1, \mathcal{T}_1) = \{\text{Professor} \sqcup \text{PostDoctor} \sqsubseteq \text{Employee},$
  $\text{PhDSupervisor} \equiv$
  $\text{Professor} \sqcap \exists \text{isSupervisorof.PhDStudent}\}$, $\text{Forget}(V_2, \mathcal{T}_2) = \mathcal{T}_2$
  It is easy to check that the DKB $\mathcal{D}_1 = \langle \{\langle \text{Forget}(V_i, \mathcal{T}_i), \mathcal{A}_i \rangle\}_{i=1,2}, \mathfrak{B} \rangle$ *is consistent.*
- $\overrightarrow{V}_2 = \langle V_1, V_2 \rangle$, *where*
  $V_1 = \varnothing$ *and* $V_2 = \{\text{PhDStudent}, \text{PhDSupervisor}\}$:
  $\text{Forget}(V_1, \mathcal{T}_1) = \mathcal{T}_1$,
  $\text{Forget}(V_2, \mathcal{T}_2) = \{\text{Professor} \sqcup \text{Lecturer} \sqsubseteq \text{AcademicStaff}, \text{Student} \sqcap \text{AcademicStaff} \sqsubseteq \bot\}$
  The DKB $\mathcal{D}_2 = \langle \{\langle \text{Forget}(V_i, \mathcal{T}_i), \mathcal{A}_i \rangle\}_{i=1,2}, \mathfrak{B} \rangle$ *is consistent.*

However, the following vector is not a recovery for $\mathcal{D}$ *given* $\mathcal{C}_\mathcal{D}$: $\overrightarrow{V} = \langle V_1, V_2 \rangle$, *where* $V_1 = \{\text{PhDStudent}\}$ *and* $V_2 = \varnothing$, *because it does not satisfy the constraints of H in* $\mathcal{C}_\mathcal{D}$.

4.3.2 Preferred Recoveries

Given a forgetting context, there may exist several different recoveries. In many cases, we are only interested in some of these recoveries, called *preferred recoveries*, which is defined by some preference relation on recoveries. For example, among all the recoveries, we may be only interested in those contain minimal number of concepts to forgotten. The notion of preferred recovery is originally defined in [LM02] in propositional logic. This is important to ensure minimal change. That is, among those recoveries, we may give preference to those that forget less concept names. This coincides with the principle of minimal change in belief revision. Actually, it has been shown in [LM02] that some model-based revision operators belong to their framework based on variable forgetting. We adapt it to distributed systems here. We need to define a preference relation on the set of all recoveries for a distributed system $\mathcal{D}$ given a forgetting context $\mathcal{C}_\mathcal{D}$ for it. For any two recoveries $\overrightarrow{V}$ and $\overrightarrow{V}'$, $\overrightarrow{V} \subseteq_c \overrightarrow{V}'$ means that $V_i \subseteq V_i'$ for each $i = 1, 2$.

Definition 27 *Given a distributed system* $\mathcal{D}$ *and a forgetting context* $\mathcal{C}_\mathcal{D}$ *for it, a preference relation* $\preceq$ *on* $\mathcal{R}_{\mathcal{C}_\mathcal{D}}(\mathcal{D})$ *is a reflexive and transitive relation which satisfies the following property:*

$$\forall \overrightarrow{V}, \overrightarrow{V}' \in \mathcal{R}_{\mathcal{C}_\mathcal{D}}(\mathcal{D}),\ if\ \overrightarrow{V} \subseteq_c \overrightarrow{V}',\ then\ \overrightarrow{V} \preceq \overrightarrow{V}'.$$

As usual, we denote $\overrightarrow{V} \prec \overrightarrow{V}'$ *for* $\overrightarrow{V} \preceq \overrightarrow{V}'$ *and* $\overrightarrow{V}' \not\preceq \overrightarrow{V}$, *and* $\overrightarrow{V} \sim \overrightarrow{V}'$ *for* $\overrightarrow{V} \preceq \overrightarrow{V}'$ *and* $\overrightarrow{V}' \preceq \overrightarrow{V}$.

In Definition 27, a recovery $\overrightarrow{V}$ is at least preferred to another one $\overrightarrow{V}'$ ($\overrightarrow{V} \preceq \overrightarrow{V}'$) if each V_i in $\overrightarrow{V}$ is a subset of V_i' in $\overrightarrow{V}'$.
There are many ways to define a preference relation. For instance, we can prefer recoveries that lead to forget minimal sets of concepts $w.r.t.$ the set containment. We can also ask the experts or users for such a preference relation. We adapt two specific preference relations in [LM02] as follows.

- *binaricity* preference relation $\preceq_{bin}$: $\forall \overrightarrow{V}, \overrightarrow{V}' \in \mathcal{R}_{\mathcal{C}_\mathcal{D}}(\mathcal{D})$, if for every $i = 1, ..., n$, $(V_i \neq \varnothing \Leftrightarrow V_i' \neq \varnothing)$, then $\overrightarrow{V} \sim_{bin} \overrightarrow{V}'$.

- *ranking function* based preference relation $\preceq_\mu$: suppose μ is a *ranking function* from $\mathcal{R}_{\mathcal{C}_\mathcal{D}}(\mathcal{D})$ to $\mathbb{N}$ such that $\mu(\langle \varnothing, ..., \varnothing \rangle) = 0$, and $\forall \overrightarrow{V}, \overrightarrow{V}' \in \mathcal{R}_{\mathcal{C}_\mathcal{D}}(\mathcal{D})$, if $\overrightarrow{V} \subseteq_c \overrightarrow{V}'$, then $\mu(\overrightarrow{V}) \leq \mu(\overrightarrow{V}')$. The preference relation $\preceq_\mu$ induced by μ is the total pre-ordering defined as follows. For all $\overrightarrow{V}, \overrightarrow{V}' \in \mathcal{R}_{\mathcal{C}_\mathcal{D}}(\mathcal{D})$, $\overrightarrow{V} \preceq_\mu \overrightarrow{V}'$ if and only if
$\mu(\overrightarrow{V}) \leq \mu(\overrightarrow{V}')$.

The ranking function based preference relation is defined by a ranking function from $\mathcal{R}_{\mathcal{C}_\mathcal{D}}(\mathcal{D})$ to $\mathbb{N}$. A particular ranking function can be defined by $\mu_{card}(\overrightarrow{V}) = |\bigcup \overrightarrow{V}|$, where $\bigcup \overrightarrow{V} = V_1 \cup ... \cup V_n$. We denote the preference relation based on μ_{card} as $\preceq_{card}$. A recovery $\overrightarrow{V}$ is preferred to another one $\overrightarrow{V}'$ $w.r.t.$ the preference relation $\preceq_{card}$ if and only if the cardinality of the union of the sets V_i $(i = 1, ..., n)$ is smaller than that of the union of the sets V_i' $(i = 1, ..., n)$.
By Corollary 1, when a concept name is forgotten in a DL knowledge base, the resulting DL knowledge base is weaker than the original one $w.r.t.$ the inference power. Given $\overrightarrow{V}, \overrightarrow{V}' \in \mathcal{R}_{\mathcal{C}_\mathcal{D}}(\mathcal{D})$, if $\overrightarrow{V} \subseteq_c \overrightarrow{V}'$ then $\mathcal{D}|\overrightarrow{V} \models \mathcal{D}|\overrightarrow{V}'$. Therefore, among all the recoveries from $\mathcal{R}_{\mathcal{C}_\mathcal{D}}(\mathcal{D})$, those which are minimal $w.r.t.$ $\subseteq_c$ lead to cures preserving as much information as possible given the forgetting context $\mathcal{C}_\mathcal{D}$.
Next, we define two consequence relations. We denote by $\overrightarrow{V}^{\preceq}_{\mathcal{D},\mathcal{C}_\mathcal{D}}$ the set of all minimal recoveries from $\mathcal{R}_{\mathcal{C}_\mathcal{D}}(\mathcal{D})$ $w.r.t.$ $\preceq$.

Definition 28 *Given a distributed system $\mathcal{D}$ with Cn as its consequence relation and a forgetting context $\mathcal{C}_\mathcal{D}$ for it, let $\preceq$ be a preference relation on $\mathcal{R}_{\mathcal{C}_\mathcal{D}}(\mathcal{D})$. Let ϕ be a DL axiom. Then ϕ is said to be sceptically inferred from $\mathcal{D}$ $w.r.t.$ $\preceq$, denoted by $\mathcal{D} \models^{\mathcal{C}_\mathcal{D}}_{\preceq,Ske} \phi$, if and only if $\mathcal{D}|\overrightarrow{V} \models \phi$ for all $\overrightarrow{V} \in \overrightarrow{V}^{\preceq}_{\mathcal{D},\mathcal{C}_\mathcal{D}}$. ϕ is called a sceptical consequence of $\mathcal{D}$.*

A DL axiom is a sceptical consequence of a distributed system $\mathcal{D}$ if and only if it can be inferred from all the most preferred recoveries.

Example 6 *(Example 5 Continued) Suppose $\preceq = \preceq_{card}$. Then $\overrightarrow{V}_1$ is a preferred recovery because $|\bigcup \overrightarrow{V}_1| = 1$ and no other recovery can have cardinality smaller than it. Other preferred recoveries are given as follows:*

- $\overrightarrow{V}_3 = \langle V_1, V_2 \rangle$*, where* $V_1 = \{\mathsf{Employee}\}$ *and* $V_2 = \varnothing$*. We have*
$\mathsf{Forget}(V_3, \mathcal{T}_1) =$
$\{\mathsf{PhDStudent} \sqsubseteq \mathsf{ResearchAssistant},$
$\mathsf{PhDSupervisor} \equiv \mathsf{Professor} \sqcap \exists \mathsf{isSupervisorof}.\mathsf{PhDStudent}\}$ *and* $\mathsf{Forget}(V_3, \mathcal{T}_2) = \mathcal{T}_2$.

- $\overrightarrow{V}_4 = \langle V_1, V_2 \rangle$*, where* $V_1 = \varnothing$ *and* $V_2 = \{\mathsf{AcademicStaff}\}$*. We have* $\mathsf{Forget}(V_4, \mathcal{T}_1) = \mathcal{T}_1$ *and*
$\mathsf{Forget}(V_4, \mathcal{T}_2) = \{\mathsf{PhDStudent} \sqsubseteq \mathsf{Student}, \mathsf{PhDSupervisor}$
$\sqsubseteq \mathsf{AcademicStaff} \sqcap \exists \mathsf{isSupervisorof}.\mathsf{PhDStudent}\}$.

- $\overrightarrow{V}_5 = \langle V_1, V_2 \rangle$*, where* $V_1 = \varnothing$ *and* $V_2 = \{\mathsf{Student}\}$*. We have* $\mathsf{Forget}(V_5, \mathcal{T}_1) = \mathcal{T}_1$ *and*
$\mathsf{Forget}(V_5, \mathcal{T}_2) = \{\mathsf{Professor} \sqcup \mathsf{SeniorLecturer} \sqcup \mathsf{Lecturer} \sqcup \mathsf{ResearchAssistant}$
$\sqsubseteq \mathsf{AcademicStaff}\}$.

Since $\mathsf{PhDSupervisor} \equiv \mathsf{Professor} \sqcap \exists \mathsf{isSupervisorof}.\mathsf{PhDStudent}$ *is in* $\mathsf{Forget}(V_i, \mathcal{T}_i)$ *for all* $i = 1, 3, 4, 5$*, we have that* $\mathcal{D}|\overrightarrow{V} \models \mathsf{Professor}(Tom)$*, for all* $\overrightarrow{V} \in \overrightarrow{V}^{\preceq}_{\mathcal{D},\mathcal{C}_\mathcal{D}}$*. That is, we can conclude that* Tom *is a professor using the sceptical inference.*

Suppose $\mathcal{D} = \langle\{O_i\}_{i\in I}, \mathfrak{B}\rangle$ is a distributed knowledge base in DDL. A maximal consistent sub-base of $\mathcal{D}$ $w.r.t$ terminology is a sub-base $\mathcal{D}' = \langle\{O_i'\}_{i\in I}, \mathfrak{B}\rangle$ (with $O' = \langle\mathcal{R}', \mathcal{T}', \mathcal{A}'\rangle$) of $\mathcal{D}$ which satisfies the following conditions: (1) $\mathcal{D}'$ is consistent; (2) $\{\mathcal{T}_i' : \mathcal{T}_i' \neq \varnothing\} \subseteq \{\mathcal{T}_i : i = 1, ..., n\}$, $\mathcal{R}' = \mathcal{R}$, and $\mathcal{A}_i' = \mathcal{A}_i$; (3) there does not exist a sub-base $\mathcal{D}''$ of $\mathcal{D}$ which satisfies (1) and (2), and $\{\mathcal{T}_i' : \mathcal{T}_i' \neq \varnothing\} \subset \{\mathcal{T}_i'' : \mathcal{T}_i'' \neq \varnothing\}$.

Proposition 9 *Let $\mathcal{D}$ be a distributed knowledge base and* MaxCons($\mathcal{D}$) *the set of all maximal consistent sub-bases of $\mathcal{D}$. Suppose $\mathcal{C}_\mathcal{D}$ is the standard forgetting context and the preference relation $\preceq$ is a binaricity preference relation. Then for every DL axiom ϕ, $\mathcal{D} \models^{\mathcal{C}_\mathcal{D}}_{\preceq,Ske} \phi$ if and only if $\mathcal{D}_i \models \phi$, for all $\mathcal{D}_i$*∈MaxCons($\mathcal{D}$).

Proposition 9 tells us that the sceptical consequence relation based on binaricity preference relation is the same as the consequence relation based on maximal consistent sub-bases $w.r.t$ terminology.

In the case that many preferred recoveries exist (see Example 5), the sceptical consequence relation is rather weak (i.e., only few DL axioms can be inferred). Therefore, we propose another consequence relation.

Definition 29 *Given a distributed system $\mathcal{D}$ with Cn as its consequence relation and a forgetting context $\mathcal{C}_\mathcal{D}$ for it, let $\preceq$ be a preference relation on $\mathcal{R}_{\mathcal{C}_\mathcal{D}}(\mathcal{D})$. Let ϕ be a DL axiom and $\neg\phi$ is its negation*[2]*. Then ϕ is said to be argumentatively inferred from $\mathcal{D}$ $w.r.t.$ $\preceq$, denoted by $\mathcal{D} \models^{\mathcal{C}_\mathcal{D}}_{\preceq,Arg} \phi$, if and only if there exists a $\overrightarrow{V} \in \overrightarrow{V}^{\preceq}_{\mathcal{D},\mathcal{C}_\mathcal{D}}$ such that $\mathcal{D}|\overrightarrow{V} \models \phi$ and there does not exist $\overrightarrow{V}' \in \overrightarrow{V}^{\preceq}_{\mathcal{D},\mathcal{C}_\mathcal{D}}$ such that $\mathcal{D}|\overrightarrow{V}' \models \neg\phi$. ϕ is called an argumentative consequence of $\mathcal{D}$.*

A DL axiom is an argumentative consequence if and only if it can be inferred from a most preferred recovery and its negation cannot be inferred from any most preferred recoveries. Therefore, the argumentative consequence relation will not result in contradictory conclusions. Note that our argumentative consequence relation is different from consequence relation defined in the logic-based framework for argumentation [BH01] because a most preferred recovery can be viewed as maximal recovered consistent information in the distributed system whilst an argument in an argumentation theory is a minimal consistent subset that can infer a conclusion. Unlike an argument in the argumentation theory, each axiom in a most preferred recovered is self-justified. A weakness of the argumentative consequence relation is that it may be too adventurous because an argumentative consequence may not be inferred by other most preferred recoveries (also its negation will not be inferred). The relationship between this consequence relation and the sceptical consequence relation is summarized by the following proposition.

Proposition 10 *Given a distributed system $\mathcal{D}$ with Cn as its consequence relation and a forgetting context $\mathcal{C}_\mathcal{D}$ for it, suppose ϕ is a DL axiom. If $\mathcal{D} \models^{\mathcal{C}_\mathcal{D}}_{\preceq,Ske} \phi$, then $\mathcal{D} \models^{\mathcal{C}_\mathcal{D}}_{\preceq,Arg} \phi$.*

That is, every sceptical consequence of a distributed system is an argumentative consequence.

Example 7 *(Example 6 Continued) It is easy to check that, for $i = 3, 4, 5$, $\mathcal{D}|\overrightarrow{V}_i \models$* ResearchAssistant($Mary$) *because* PhDStudent($Mary$)∈$\mathcal{A}_1$ *and* PhDStudent $\sqsubseteq$ ResearchAssistant∈Forget($V_i, \mathcal{T}_1$). *Since* ResearchAssistant$\notin$ Con(Forget($V_1, \mathcal{T}_1$)) *and* ResearchAssistant$\notin$Con($\mathcal{T}_2$), *we must have that $\mathcal{D}|\overrightarrow{V}_1 \not\models \neg$*ResearchAssistant($Mary$). *Therefore,* $\mathcal{D} \models^{\mathcal{C}_\mathcal{D}}_{\preceq,Arg}$ ResearchAssistant($Mary$).

[2]The notion of axiom negation is defined in [FHP+06]

Chapter 5

Conclusion

5.1 Summary

In this deliverable we proposed some inconsistency-tolerant approach for reasoning with networked ontologies. We first extended the semantics of description logic $\mathcal{ALC}$ with a four-valued semantics. We implemented an algorithm for reasoning with the four-valued semantics and provide a prototype called Inconsistency Handler. The idea of the algorithm is to transform inconsistent OWL ontologies into non-contradictory knowledge bases. With Inconsistency Handler as a plug-in, Neon Toolkit can work using 4-valued logic, without the need to develop a new inference machine. We then proposed a bilattice-based semantics to generalize the four-valued semantics. The bilattice-based semantics is very useful when we want to integrate networked ontologies. We proposed an approach for obtaining bilattices. We extended $\mathcal{SROIQ}$, the description logic underlying the proposed OWL2 [GM08], to $\mathcal{SROIQ} - \mathcal{T}$ evaluated on a logical bilattice. The bilattice-based semantics can be used to reasoning with trust information and deal with inconsistency.

Both the four-valued semantics and the bilattice-based semantics are defined for a single ontology, we propose another approach for reasoning with distributed ontologies which is based on concept forgetting. We first defined the notion of concept forgetting in DL $\mathcal{ALC}$. Some properties of the concept forgetting were given. The definitions of recoveries and preferred recoveries in propositional logic setting [LM02] were adapted to distributed systems. Based on the preferred recoveries, two consequence relations on an inconsistent distributed system were defined. One is called sceptical consequence and the other is called argumentative consequence. We showed that every sceptical consequence of a distributed system is an argumentative consequence.

5.2 Roadmap

We have described some approaches for reasoning with networked ontologies. However, some of the approaches are not fully implemented and evaluated. For example, there is no algorithm for bilattice-based approach and there is no algorithm for the forgetting-based approach. In our future work, we will provide algorithms for these inconsistency-tolerant approaches and give evaluation results for their implementation.

Bibliography

[AA96] O. Arieli and A. Avron. Reasoning with logical bilattices. *Journal of Logic, Language and Information*, 5:25–63, 1996.

[AA98] O. Arieli and A. Avron. The value of the four values. *Artificial Intelligence*, 102:97–141, 1998.

[BCM+03] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, New York, NY, USA, 2003.

[Bel77] N.D. Belnap. A useful four-valued logic. *Modern uses of multiple-valued logics* , pages 7–73, 1977.

[BH01] Philippe Besnard and Anthony Hunter. A logic-based theory of deductive arguments. *Artif. Intell.*, 128(1-2):203–235, 2001.

[BS03] Alexander Borgida and Luciano Serafini. Distributed description logics: Assimilating information from peer sources. *J. Data Semantics*, 1:153–184, 2003.

[CM07] Andrei Tamilin Christian Meilicke, Heiner Stuckenschmidt. Repairing ontology mappings. In *Proc. of AAAI'07*, pages 1408–1413. 2007.

[ES07] Jerome Euzenat and Pavel Shvaiko. *Ontology Matching*. Berlin; Heidelberg, Springer, 2007.

[FHP+06] Giorgos Flouris, Zhisheng Huang, Jeff Z. Pan, Dimitris Plexousakis, and Holger Wache. Inconsistencies, negations and changes in ontologies. In *Proc. of AAAI'06*, pages 1295–1300, 2006.

[Fit02] Melvin Fitting. Fixpoint Semantics for Logic Programming - A Survey. *Theoretical Computer Science*, 278(1-2), 2002.

[Gin92] Matthew L. Ginsberg. Multivalued Logics: A Uniform Approach to Inference in Artificial Intelligence. *Computational Intelligence*, 4(3), 1992.

[GM08] Bernardo Cuenca Grau and Boris Motik. OWL 2 Web Ontology Language: Model-Theoretic Semantics. http://www.w3.org/TR/owl2-semantics/ [2008-05], 2008.

[HBP+07] Peter Haase, Saartje Brockmans, Raul Palma, Jérôme Euzenat, and Mathieu d'Aquin. Updated version of the networked ontology model. Technical Report D1.1.2, University of Karlsruhe, AUG 2007.

[HM05] P. Haase and B. Motik. A Mapping System for the Integration of OWL-DL Ontologies. In *In Proceedings of the ACM-Workshop: Interoperability of Heterogeneous Information Systems (IHIS05)*, November 2005.

[HPS04] Ian Horrocks and Peter Patel-Schneider. Reducing OWL entailment to description logic satisfiability. *J. of Web Semantics*, 1(4):345–357, 2004.

[Hun06] A. Hunter. How to act on inconsistent news: Ignore, resolve, or reject. *Data Knowl. Eng.*, 57(3):221–239, 2006.

[HvHH+05] Peter Haase, Frank van Harmelen, Zhisheng Huang, Heiner Stuckenschmidt, and York Sure:. A framework for handling inconsistency in changing ontologies. In *Proc. of 4th International Semantic Web Conference (ISWC'05)*, pages 353–367. Springer, 2005.

[HvHtT05] Zhisheng Huang, Frank van Harmelen, and Annette ten Teije. Reasoning with inconsistent ontologies. In *Proc. of 19th International Joint Conference on Artificial Intelligence(IJCAI'05)*, pages 254–259. Morgan Kaufmann, 2005.

[KLWW08] B. Konev, C. Lutz, D. Walther, and F. Wolter. Formal properties of modularisation. In H. Stuckenschmidt and S. Spaccapietra, eds., *Ontology Modularization,* Springer, 2008.

[LM02] Jérôme Lang and Pierre Marquis. Resolving inconsistencies by variable forgetting. In *Proc. of KR'02*, pages 239–250, 2002.

[LR94] F. Lin and R. Reiter. Forget it! In *AAAI Fall Symposium on Relevance*, 1994.

[MLL06] Yue Ma, Zuoquan Lin, and Zhangang Lin. Inferring with inconsistent OWL DL ontology: A multi-valued logic approach. In *Proc. of EDBT Workshops*, pages 535–553, 2006.

[MS07] Christian Meilicke and Heiner Stuckenschmidt. Applying logical constraints to ontology matching. In *Proc. of KI'07*, pages 99–113, 2007.

[P. 05] P. Haase et al. A Framework for Handling Inconsistency in Changing Ontologies. In *Proc. of ISWC*, 2005.

[PS89] Peter F. Patel-Schneider. A four-valued semantics for terminological logics. *Artificial Intelligence*, 38:319–351, 1989.

[PSH04] P.F. Patel-Schneider and Ian Horrocks. OWL web ontology language semantics and abstract syntax. W3C Recommendation, 2004.

[QHJ07] Guilin Qi, Peter Haase, and Qiu Ji. D1.2.1 consistency models for networked ontologies. Technical Report D1.2.1, Universität Karlsruhe, FEB 2007.

[QHJ08] Guilin Qi, Peter Haase, and Qiu Ji. D1.2.3 diagnosing and repairing inconsistent networked ontologies. Technical Report D1.2.3, Universität Karlsruhe, September 2008.

[QHJV07] Guilin Qi, Peter Haase, Qiu Ji, and Johanna Voelker. D1.2.2 consistency models for networked ontologies–evaluation. Technical Report D1.2.2, Universität Karlsruhe, September 2007.

[SC03] Stefan Schlobach and Ronald Cornet. Non-standard reasoning services for the debugging of description logic terminologies. In *Proc. of IJCAI'03*, pages 355–362, 2003.

[ST05] L. Serafini and A. Tamilin. Drago: Distributed reasoning architecture for the semantic web. In *Proceedings of the Second European Semantic Web Conference, ESWC 2005, Heraklion, Crete, Greece, May 29 - June 1, 2005*, pages 361–376, 2005.

[Str97] Umberto Straccia. A sequent calculus for reasoning in four-valued description logics. In *Proc. of TABLEAUX'97*, pages 343–357, 1997.

[Str06] Umberto Straccia. A Fuzzy Description Logic for the Semantic Web. In *Fuzzy Logic and the Semantic Web*. Elsevier, 2006.

[WWTP08] Zhe Wang, Kewen Wang, Rodney W. Topor, and Jeff Z. Pan. Forgetting concepts in dl-lite. In *Proc. of ESWC'08*, pages 245–257, 2008.

[ZD08] Antoine Zimmermann and Chan Le Duc. Reasoning with a network of aligned ontologies. In *Proc. of RR'08*, pages 43–57, 2008.

[ZE06] Antoine Zimmermann and Jérôme Euzenat. Three semantics for distributed systems and their relations with alignment composition. In *Proc. of ISWC'06*, pages 16–29, 2006.