FOUR-VALUED LOGICS FOR PARACONSISTENT REASONING

—

Diplomarbeit von Andreas Christian Lang

Betreuender Hochschullehrer: Prof. Dr. Steffen Hölldobler
Betreuer: Dr. habil. Pascal Hitzler,
M.Sc. Markus Krötzsch

INSTITUT FÜR KÜNSTLICHE INTELLIGENZ
TECHNISCHE UNIVERSITÄT DRESDEN

Dresden, März 2006

i

Abstract

A drawback of many logics for reasoning on Semantic Web ontologies is the Principle of Explosion in case of inconsistent data. Considering a four-valued semantics, inconsistencies can be tolerated such that paraconsistency is obtained. We extend Belnap's four-valued logic to a four-valued first-order logic with implication and equality, and we develop a method for translating formulas to a representation in classical first-order logic. This procedure allows paraconsistent reasoning in classical first-order logic. We transfer our results to a setting of Description Logics by defining four-valued versions of $\mathcal{ALC}$ and $\mathcal{SHIQ}$. This lays the foundations for implementing paraconsistent reasoning for KRR systems that work with classical logics. We exemplify a realisation with respect to the ontology management infrastructure KAON2. The crucial idea is enabling KAON2 to merge ontologies and to perform reasoning tasks on the resulting ontology even if this ontology is inconsistent.

ii

"Ja, ich sage schon jetzt voraus: es werden mathematische Untersuchungen über Kalküle kommen, die einen Widerspruch enthalten, und man wird sich noch etwas darauf zugute tun, daß man sich auch von der Widerspruchsfreiheit emanzipiert."

Ludwig Wittgenstein, 1930.[1]

---

[1][Wit81, Page 332]

iv

Contents

Chapter 1

Introduction

Traditionally, the consensus of opinion among designers of information systems is that inconsistency is undesirable. Inconsistent information may cause a system to draw erroneous conclusions which is against the purpose of an information system. However, inconsistencies arise continuously in real world situations. For example, asking a German about the nationality of the writer Franz Kafka, you might get the answer that Franz Kafka is German. Asking a Czech concerning this matter, she may answer that Franz Kafka is not German but Czech. Finally, some historian would likely answer that Franz Kafka was Austro-Hungarian and neither German nor Czech. Now, how do we suppose an information system to act in this situation? Should it refuse certain information to avoid the inconsistency even if it does not exhibit a criterion for deciding which information is true?

We take another approach by analysing at first why an inconsistency involves such big trouble. The problem that we have is rooted in the idea that logic is bivalent. This means, concerning our example, Franz Kafka cannot be a German fellow countryman and *not* a German fellow countryman simultaneously. Confronted with such a contradiction, an information system based on classical logic becomes useless for any reasoning task. This is due to the fact that classical logic allows to conclude anything on basis of an inconsistency. In literature, this principle is called the *Principle of Explosion* (as well known as *ex contradictione sequitur quodlibet*). If we confront a human to the above contradiction we observe that commonsense reasoning works differently. Even if a human encounters an inconsistency, nothing hinders her to accept this fact and to draw valid conclusions about other things. Such a behaviour is an important feature of a robust information and reasoning system. We want a system to be able to use distributed information sources and we want it to be competent to perform reasoning tasks on the joined information. At the same time, nobody guarantees for the consistency of the joined information. It occurs too often that information is forged because of subjective interpretations of humans, or just because the information is incomplete or imprecise. The chance for inconsistencies becomes larger the more different sources and information we regard. In order to scale up an information system's capacity, it is essential to handle inconsistencies adequately.

Currently, information is commonly formalised using classical logic and many work was spend in the development of sophisticated information systems that work with classical logic. Thus, we would like to use existing systems in future and just enhance their robustness. In this thesis, we provide a technique to achieve a major robustness that allows reasoning on inconsistent information – so-called paraconsistent reasoning. On the one hand, we develop in detail the theoretical foundations, and, on the other hand, we discuss how the theory may be put into practice with respect to KAON2[1] – an information system developed in the University of Karlsruhe. The idea is to enable the system to merge information from various sources and to perform reasoning tasks based on the combined information that may contain inconsistencies.

Idea of Paraconsistency

The idea of paraconsistency is rooted in the desire to perform reasoning on inconsistent information. Consequently, a paraconsistent logic rejects the Principle of Explosion. This means with respect to the above example of an inconsistency, viz "Franz Kafka is a German fellow countryman and Franz Kafka is not a German fellow countryman", that our information system tolerates the contradiction. Tolerating the contradiction does not mean to ignore the information. In this simple case, one could argue to remove all information about the nationality of Franz Kafka but this destroys a large part of the knowledge and likewise the inferential capabilities of the information system. Furthermore, since one source told our information system that Franz Kafka is German, there exist evidence that Franz Kafka is German. The same is valid for the contrary and we should respect that there is evidence for both and, thus, we should not simply delete the information. Of course, we cannot assume the information system to draw valid conclusions concerning the nationality of Franz Kafka but it can conclude that different opinions about the nationality of Franz Kafka exist. As pointed out in [WOS03], if we consider knowledge representation not as a representation of true believes but as representation of pure information, then it is reasonable to treat positive and negative information about a certain issue as independent.

Sometimes it might be welcome to derive that a contradiction exists with respect to a certain matter. Imagine cases where contradictions are an inherent property of the problem, so-called *dialetheias*. A well known example is the "Liar's Paradox" where a sentence states its own falsity ("This sentence is not true"). Information systems that combine data from different sources should be able to handle those cases [SBB+05].

Several methods were proposed to obtain paraconsistency such as non-adjunctive logics [Jas69], non-truth-functional logics [dC74], relevance logics [DR86, AB90], and many-valued logics [Bel76, Bel75, AT75, Ase66]. In this thesis, we follow the approach of Belnap, who developed a four-valued logic [Bel76, Bel75]. A four-valued logic facilitates paraconsistency in an intuitive way by offering additional truth values. This means with respect to the sen-

---

[1]http://kaon2.semanticweb.org

tence "Franz Kafka is a German fellow countryman and Franz Kafka is not a German fellow countryman" that we assign a different truth value than *false* to it[2]. Typically, a four-valued logic provides a special truth value for contradictory information such that explosion is avoided in the presence of a contradiction.

The reader interested in philosophical aspects of paraconsistency is referred to [Béz99, Tan03]. In [Béz99], Béziau elucidates various ideas of paraconsistency. He sets paraconsistent logics in relation to other logics and describes their applications. In [Tan03], Tanaka compares different philosophical positions concerning paraconsistent reasoning.

The reader interested in more technical details is referred to [Pri02, Hun98, dCKB04, dC74, dCBB95, Res03, Béz02, Béz98, AA98, Bel76, Bel75]. In [Pri02, Hun98], Priest and Hunter give a general introduction into paraconsistent logics. In [dCKB04], da Costa, Krause, and Bueno give an overview about several approaches to paraconsistency including a detailed introduction to da Costa's $\mathcal{C}$-logics as well as a description of current developments and applications of paraconsistent logics. In [Res03], Restall shows that actually many non-classical logics behave paraconsistent because inconsistencies are tolerated. In [Béz02], Béziau illustrates that paraconsistent behaviour can be obtained in the modal logic $S5$ and that the latter can be represented in classical first-order logic. In the following, we refer especially to [AA98] and [Bel76, Bel75], where a valuable basis for paraconsistent reasoning with four-valued logics is elaborated.

Outline of the Thesis

This thesis comprises the further development of the logical foundations for paraconsistent reasoning. A central aspect is the elaboration of a four-valued first-order logic with an implication connective that allows to express logical consequences as formulas of the logic. Algorithms are developed that translate formulas of this four-valued first-order logic to a setting of two-valued first-order logic such that paraconsistent reasoning becomes possible in classical first-order logic. Furthermore, we transfer the results from four-valued first-order logic to four-valued Description Logics which allows to perform paraconsistent reasoning by reasoning systems for classical Description Logics. As application example, we refer to the ontology management infrastructure KAON2 and we discuss how an implementation may be realised for this system.

The crucial idea of our work is to refer to existing ontologies[3] and to merge them in order to enlarge the knowledge. These ontologies are encoded in classical logic but we assign a four-valued semantics such that inconsistent ontologies become paraconsistent. These paraconsistent ontologies are translated to a consistent representation in a classical logic. This enables KAON2 to merge the resulting ontologies even if the original set union is inconsistent.

After this introductory chapter, the thesis is organised as follows:

---

[2]The option to assign the truth value *true* is obviously not passable.

[3]For a definition of the term *ontology* we refer to Chapter 5.2 and [SWM04]. For the moment it suffices to regard an ontology as a source of information.

Chapter 2 (Logical Foundations). The considered logical foundations in this chapter cover classical first-order logic, the Description Logics $\mathcal{ALC}$ and $\mathcal{SHIQ}$, and an introduction of the formal aspects of paraconsistency by a four-valued semantics. In the latter section, we elucidate additionally the relation of four-valued logics to logical bilattices which gives an intuition how paraconsistent reasoning works. The notation that we introduce in this chapter is used throughout the entire work and the introduced definitions are assumed to be known in later chapters.

Chapter 3 (Four-valued First-order Logic). In this chapter, we extend Belnap's four-valued logic by an appropriate implication connective. We give a detailed presentation of the formal aspects of paraconsistent reasoning and we show that paraconsistent reasoning can be performed in classical first-order logic. To this end, we develop a method for mapping four-valued first-order logic formulas to classical first-order logic formulas and we prove the correctness of this proceeding. The crucial algorithms for obtaining paraconsistent reasoning capabilities in a classical first-order logic are presented in detail. Finally, we lay the foundations for transferring the result to a setting of Description Logics.

Chapter 4 (Four-valued Description Logic). With this chapter, we transfer the previously developed formalism for paraconsistent reasoning to a setting for Description Logic. First, we propose a four-valued version of $\mathcal{ALC}$, called $\mathcal{ALC}4$, and second, we discuss an extension of $\mathcal{ALC}4$ to four-valued $\mathcal{SHIQ}$, i.e., $\mathcal{SHIQ}4$. We give the algorithms for translating $\mathcal{ALC}4$ and $\mathcal{SHIQ}4$ to $\mathcal{ALC}$ and $\mathcal{SHIQ}$, respectively, such that paraconsistent reasoning becomes possible in classical Description Logics.

Chapter 5 (The Realisation in KAON2). This chapter describes how the theory might be applied in practice. We give a brief introduction into OWL/XML ontologies and elucidate how the previously developed algorithms can be built into the ontology management infrastructure KAON2. We conclude the chapter with an example that illustrates the extended reasoning capabilities of KAON2.

Chapter 6 (Conclusions and Related Work). This last chapter gives a résumé of the thesis and provides an outlook for future developments. We set our work in relation to other approaches in this field and we give an overview about the most important approaches.

Conventions

Throughout this thesis we specify the general term "information system" as *Knowledge Representation and Reasoning System* that we abbreviate by the term "KRR system".

With respect to logical connectives, we define that unary connectives have a higher precedence than binary connectives and unary connectives have the same precedence among each other. For binary connectives, we determine that

the precedence of a conjunction and disjunction is higher than the precedence of implication and equivalence. Where it is possible, we omit writing outermost parentheses of formulas.

Prerequisites

We expect the reader to possess a basic knowledge about fundamental mathematical terms. Thus, we presume the definitions of a *partial order*, a *partially ordered set*, *infimum* (*meet*), *supremum* (*join*), *lattice*, and *ordering* to be known.

Chapter 2

Logical Foundations

Within the following three sections, we lay the logical foundations for this thesis. Important terms are introduced and principle definitions are given to which we refer in later chapters.

The first section gives a brief introduction into classical first-order logic. The self-confident reader familiar to classical first-order logic may skip this section safely. However, we include it for the purpose of self-containedness of this thesis and because the notation that we use throughout the work is explained in a common setting.

The second section gives an overview about Description Logics and introduces $\mathcal{ALC}$ and $\mathcal{SHIQ}$. In Chapter 4, we refer to this second section but the reader that is familiar to $\mathcal{ALC}$ and $\mathcal{SHIQ}$ may skip this second section.

The third section covers the formal aspects of paraconsistency and contains important definitions required in later Chapters. Since these definitions are not standard definitions, we recommend reading this third section. Furthermore, the notion of a four-valued semantics is introduced and the relation of this semantics to logical bilattices is elucidated. We refer to this semantics very closely in Chapter 3.

2.1   Classical First-order Logic

A comprehensive introduction to classical first-order logic is given in [Höl03, Fit96]. In this thesis we refer closely to [Höl00]. We introduce the most important ideas beginning with the syntax of classical first-order logic.

Syntax

The *alphabet* $\mathcal{A}$ of first-order logic consists of the following disjoint sets of symbols:

1. the set $\mathcal{A}_{\mathcal{V}}$ of *variables* of the form x, y, z;

2. the set $\mathcal{A}_{\mathcal{C}}$ of *constant symbols* of the form a, b, c;

3. the set $\mathcal{A}_{\mathcal{F}}$ of *function symbols* of the form f, g;

4. the set $\mathcal{A}_{\mathcal{R}}$ of *predicate* or *relation symbols* of the form R, S;

5. the set of *connectives* $\{\neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$;

6. the set of *quantifiers* $\{\exists, \forall\}$;

7. the set of *special symbols* $\{\top, \bot\}$;

8. and the *punctuation symbols* "(", ")", and ",".

Unless otherwise stated, we refer throughout the entire thesis to variables with the letters x, y, z, to constants with the letters a, b, c, to function symbols with the letters f and g, and to relation symbols with the letters R and S. For all letters, we allow optional sub- and superscripts.

Each function and relation symbol has a fixed *arity* n which denotes the number of arguments associated with it. The set $\mathcal{A}_{\mathcal{R}}$ of a first-order logic with equality comprises the infix written binary relation symbol $\approx$. *Terms* of a first-order logic are inductively defined as follows:

1. every variable is a term,

2. every constant symbol of $\mathcal{A}_{\mathcal{C}}$ is a term,

3. if $f \in \mathcal{A}_{\mathcal{F}}$ of arity n and $t_1, ..., t_n$ are terms, so is $f(t_1, ..., t_n)$ a term.

For notational convenience, we abbreviate the writing of a tuple $(t_1, ..., t_n)$ of terms by a bold letter $\boldsymbol{t}$. A *subterm* of a term t is a sub-string of t which is a term. *Formulas* are inductively defined as follows:

1. $\top$ and $\bot$ are formulas;

2. if R is an n-ary relation symbol and $t_1, ..., t_n$ are terms, then $R(t_1, ..., t_n)$ is an *atomic formula* (also called *atom*);

3. if $\approx \in \mathcal{A}_{\mathcal{R}}$ and t_1 and t_2 are terms, then $t_1 \approx t_2$ is a formula;

4. if F and G are formulas and x is a variable, then $\neg F$, $F \wedge G$, $F \vee G$, $F \rightarrow G$, $F \leftrightarrow G$, $(\exists x)F$, and $(\forall x)F$ are formulas.

For notational convenience, we abbreviate null-ary atoms $R()$ by R and formulas of the form $(\exists x_1), ..., (\exists x_n)F$ and $(\forall x_1), ..., (\forall x_n)F$ by $(\exists x_1, ..., x_n)F$ and $(\forall x_1, ..., x_n)F$, respectively. A *subformula* of a formula F is a sub-string of F that is a formula. Considering a set of subformulas, we call a subformula *maximal* if it does not occur as subformula of another formula in this set. *Literals* are atomic formulas, negated atomic formulas, $\top$, and $\bot$. We refer with the term *negative literal* to a negated atomic formula and with the term *positive literal* to an atomic formula that is not negated. The *language* $\mathcal{L}$ of first-order logic is an infinite set of formulas.

Unless otherwise stated, we refer throughout the entire thesis to formulas with the letters F, G, and H and we refer to sets of formulas with the letters $\mathcal{E}$, $\mathcal{F}$, $\mathcal{G}$, and $\mathcal{L}$. We allow optional sub- and superscripts.

Semantics

A *pre-interpretation* J for a language $\mathcal{L}$ based on the alphabet $\mathcal{A}$ consists of a non-empty set $\mathcal{D}$ called the *domain* of J, and a mapping ^{J} that assigns

1. to each constant symbol $c \in \mathcal{A}_\mathcal{C}$ an element c^J in the domain $\mathcal{D}$;

2. to each n-ary function symbol $f \in \mathcal{A}_\mathcal{F}$ an n-ary function $f^J : \mathcal{D}^n \to \mathcal{D}$ over the domain $\mathcal{D}$.

A *state* σ over the domain $\mathcal{D}$ of a pre-interpretation is a mapping assigning an element of $\mathcal{D}$ to each variable occurring in $\mathcal{A}_\mathcal{V}$. The meaning of a term t with respect to a state σ and a pre-interpretation J is defined as follows:

$$t^{J,\sigma} \stackrel{def}{=} \begin{cases} \sigma(t) & \text{if } t \text{ is a variable,} \\ c^J & \text{if } t \text{ is a constant } c, \\ f^J(t_1^{J,\sigma}, ..., t_n^{J,\sigma}) & \text{if } t \text{ is of the form } f(t_1, ..., t_n). \end{cases}$$

We denote with $\sigma\{x \mapsto d\}$ the state obtained from σ by assigning d to x while leaving all other assignments unchanged.

$$\sigma\{x \mapsto d\}(y) \stackrel{def}{=} \begin{cases} d & \text{if } x = y, \\ \sigma(y) & \text{if } x \neq y. \end{cases}$$

An *interpretation* I for a language $\mathcal{L}$ based on an alphabet $\mathcal{A}$ consists of a pre-interpretation J with a domain $\mathcal{D}$ and a mapping ^{I} that assigns to each n-ary relation symbol R a set $R^I \subseteq \mathcal{D}^n$. This provides the possibility to assign unambiguously a truth value to an instance $R(\boldsymbol{t})$. If $\boldsymbol{t}^{I,\sigma} \in R^I$, then we say that I *assigns* the truth value *true* (denoted by t) to $R(\boldsymbol{t})$, and if $\boldsymbol{t}^{I,\sigma} \notin R^I$, then we say that I assigns the truth value *false* (denoted by f) to $R(\boldsymbol{t})$. I is said to be *based* on J and for uniformity reasons, the mapping ^{J} is denoted by ^{I}. $\mathcal{D}$ is said to be the domain of I as well. Thus, for the rest of the thesis we refer only to interpretations and we presume the pre-interpretation. When we give the semantics in Chapter 3.2, we refer directly to an interpretation I and we define the interpretation's mapping ^{I} such that it covers the assignments 1 and 2 of the pre-interpretation's mapping as well.

With respect to an interpretation I and a state σ of a language $\mathcal{L}$, we define for all formulas F the relation $I, \sigma \models F$ as follows:

1. $I, \sigma \models \top$
2. $I, \sigma \not\models \bot$
3. $I, \sigma \models R(\boldsymbol{t})$ iff $\boldsymbol{t}^{I,\sigma} \in R^I$,
4. $I, \sigma \models \neg F$ iff $I, \sigma \not\models F$,
5. $I, \sigma \models F_1 \wedge F_2$ iff $I, \sigma \models F_1$ and $I, \sigma \models F_2$,
6. $I, \sigma \models F_1 \vee F_2$ iff $I, \sigma \models F_1$ or $I, \sigma \models F_2$,
7. $I, \sigma \models F_1 \to F_2$ iff $I, \sigma \not\models F_1$ or $I, \sigma \models F_2$,
8. $I, \sigma \models F_1 \leftrightarrow F_2$ iff $I, \sigma \models F_1 \to F_2$ and $I, \sigma \models F_2 \to F_1$,
9. $I, \sigma \models (\exists x)F$ iff $I, \sigma\{x \mapsto d\} \models F$ for some $d \in \mathcal{D}$,
10. $I, \sigma \models (\forall x)F$ iff $I, \sigma\{x \mapsto d\} \models F$ for all $d \in \mathcal{D}$,

where $R(\boldsymbol{t})$ is an atomic formula. Since all interpretations assign for arbitrary states the truth value t to $\top$ and the truth value f to $\bot$, we call $\top$ the *Truth Formula* and $\bot$ the *False Formula*.

The relation $\models$ is called *entailment relation* and we denote a first-order logic by a triple $\langle \mathcal{A}, \mathcal{L}, \models \rangle$, where $\mathcal{A}$ is the alphabet, $\mathcal{L}$ is the language, and $\models$ is the entailment relation of the logic.

According to the assignment of truth values to relation symbols, we extend the interpretation's mapping ^{I} such that it covers entire formulas. The truth value of a formula F is denoted by $F^{I,\sigma}$. Whenever we refer to arbitrary states we abbreviate $F^{I,\sigma}$ by F^I. An interpretation I and a state σ *assign* a truth value to a formula with respect to $\models$ as specified by the following scheme:

1. $F^{I,\sigma} = t$ iff $I, \sigma \models F$,

2. $F^{I,\sigma} = f$ iff $I, \sigma \not\models F$.

Equivalently, we say that an interpretation and a state *map* a formula to a truth value and, referring to the formula F, we say that F *takes* a truth value.

A formula F is *satisfiable* iff there exist an interpretation I and a state σ such that $I, \sigma \models F$. In this case, we say that I *satisfies* F with respect to σ and that I, σ is a *model* of F. Furthermore, I, σ is a model of a set $\mathcal{F}$ of formulas iff I, σ is a model of all formulas in $\mathcal{F}$. We call two formulas F and G *semantically equivalent* and we write $F \equiv G$ if we find for all interpretations I and for all states σ that:

$$F^{I,\sigma} = G^{I,\sigma}.$$

Definition 2.1.1
A set of formulas $\mathcal{F}$ *entails* a formula F (written $\mathcal{F} \models F$) iff all models of $\mathcal{F}$ are models of F too. Concerning two sets $\mathcal{F}$ and $\mathcal{G}$ of formulas, we say that $\mathcal{F} \models \mathcal{G}$ iff:

If I, σ is a model of $\mathcal{F}$, then I, σ is a model of *some* formula of $\mathcal{G}$.

Unless otherwise stated, we refer throughout the entire thesis with the letter I to an interpretation, with the letter $\mathcal{D}$ to a domain of an interpretation, with ^{I} to an interpretation's mapping, with the letter σ to a state. We allow optional sub- and superscripts for I, $\mathcal{D}$, ^{I}, and σ.

2.2   Description Logic

In [BCM$^+$03] a comprehensive overview about Description Logics is given that covers the theory, the implementation, and the applications. With the following general overview about Description Logics and the introduction of $\mathcal{ALC}$, we adhere strongly to [BCM$^+$03]. For the introduction of $\mathcal{SHIQ}$, we stick to [HS98, HMS04]. The reader interested in reasoning in $\mathcal{SHIQ}$, is also referred to [HST00, IH00].

Description Logics are a family of knowledge representation formalisms using the following three basic building blocks to model relevant domain knowledge.

1. *Individuals* describe the elements in the domain. For instance, "*franzKafka*" as a person about which we want to express knowledge.

2. *Roles* encode relations between elements of the domain. For instance, "*AuthorOfBook*" expressing that an individual is the author of a book.

3. *Concepts* are collections of elements of the domain that share the same property. For instance, the concept "*German*" that comprises all those individuals that are German.

Description Logics enable a KRR system to infer implicitly represented knowledge from the knowledge contained explicitly in its knowledge base. With respect to Description Logics, a *knowledge base* $\mathcal{KB}$ is defined to comprise two components. The first is a so-called *TBox* and the second is a so-called *ABox*. The TBox defines the *terminology*, i.e., the vocabulary of the domain. The ABox contains *assertions* about individuals of the domain that are represented in terms of the vocabulary of the TBox. The vocabulary consists of concepts and roles. The TBox is used to define complex descriptions comprising several concepts.

The representation of knowledge is one ability of a KRR system. Another ability is to perform reasoning tasks on the knowledge. A typical reasoning task is to determine satisfiability of a description, or to compute whether a description is more general than another one, i.e., whether the first description *subsumes* the second description.

In the following we give a concrete introduction into one of the most common Description Logics, namely $\mathcal{ALC}$. This Description Logic is a well known and frequently used Description Logic. Therefore, we take $\mathcal{ALC}$ as basis for developing our four-valued Description Logic. After the introduction of the standard reasoning tasks we introduce an extension of $\mathcal{ALC}$, namely $\mathcal{SHIQ}$. The Description Logic $\mathcal{SHIQ}$ is a more expressive Description Logic than $\mathcal{ALC}$ and our introduction is just thought to provide a summary of syntax and semantics of the extensions. This summary suffices to follow the discussion in Chapter 4.3 about extending our four-valued version of $\mathcal{ALC}$ to a four-valued version of $\mathcal{SHIQ}$. However, the reader that is only interested in the basic application of paraconsistent reasoning for Description Logics may skip the section about $\mathcal{SHIQ}$.

Syntax of $\mathcal{ALC}$

The basic syntactic building blocks of $\mathcal{ALC}$ are *atomic concepts* (unary predicates), *atomic roles* (binary predicates), and *individuals* (constants). We refer with the symbols $\top$ and $\bot$ to the top and bottom concept, with the capital letter A to atomic concepts, with the capital letters C and D to concepts, with the capital letter R to roles and with the lower case letters a and b to constants. Throughout the thesis, we allow optional sub- and superscripts for all letters.

Furthermore, we use the connectives $\sqcap, \sqcup, \neg, \sqsubseteq$, and $\equiv$, the quantifiers $\forall$ and $\exists$, and the punctuation symbols ",", ".", "(", and ")". The notion of a *concept* is defined inductively as follows:

1. $\top$ and $\bot$ are concepts;

2. each atomic concept A is a concept;

3. if C and D are concepts, then $\neg C$ and $(C \sqcap D)$, $(C \sqcup D)$ are concepts;

4. if C is a concept and R is a role then $\exists R.C$ and $\forall R.C$ are concepts.

Instantiated concepts $C(a)$ and instantiated roles $R(a, b)$ are called *assertions*. An *ABox* is a set of such assertions. With the term *definition*, we refer to expressions of the form $A \sqsubseteq C$ and $A \equiv C$ and with the term *terminological axiom*, we refer to expressions of the form $C \sqsubseteq D$ and $C \equiv D$. A set $\mathcal{TB}$ of definitions is called a *TBox* if every atomic concept A in $\mathcal{TB}$ occurs at most once on the left hand side of a definition. An $\mathcal{ALC}$ *knowledge base* $\mathcal{KB}$ is a tuple $(\mathcal{TB}, \mathcal{AB})$, where $\mathcal{TB}$ is a TBox and $\mathcal{AB}$ is an ABox.

Standard semantics of $\mathcal{ALC}$

A formal semantics of $\mathcal{ALC}$ concepts is given by an interpretation I that comprises a non-empty set $\mathcal{D}$ (the domain of the interpretation) and an interpretation function ^{I}. The interpretation function assigns to every atomic concept A a set $A^I \subseteq \mathcal{D}$ and to every atomic role R a binary relation $R^I \subseteq \mathcal{D} \times \mathcal{D}$. The interpretation function is extended to non-atomic concepts in the following inductive way:

$$
\begin{aligned}
\top^I &\overset{def}{=} \mathcal{D}, \\
\bot^I &\overset{def}{=} \emptyset, \\
(\neg C)^I &\overset{def}{=} \mathcal{D} \backslash C^I, \\
(C \sqcap D)^I &\overset{def}{=} C^I \cap D^I, \\
(C \sqcup D)^I &\overset{def}{=} C^I \cup D^I, \\
(\forall R.C)^I &\overset{def}{=} \{a \in \mathcal{D} \mid \forall b.(a, b) \in R^I \text{ implies } b \in C^I\}, \\
(\exists R.C)^I &\overset{def}{=} \{a \in \mathcal{D} \mid \exists b.(a, b) \in R^I \text{ and } b \in C^I\}.
\end{aligned}
$$

A concept C is *satisfiable* if there exists an interpretation I such that $C^I \neq \emptyset$. In this case we say that I *satisfies* (*is a model of*) C. A concept D *subsumes* another concept C (written $C \sqsubseteq D$) if $C^I \subseteq D^I$ for all interpretations I.

Two concepts C and D are *equivalent* (written $C \equiv D$) if $C^I = D^I$ for all interpretations I. The following equivalences are valid in $\mathcal{ALC}$:

$$
\begin{aligned}
\neg\top &\equiv \bot, \\
\neg\bot &\equiv \top, \\
\neg\neg C &\equiv C, \\
\neg(C \sqcup D) &\equiv (\neg C \sqcap \neg D), \\
\neg(C \sqcap D) &\equiv (\neg C \sqcup \neg D), \\
\neg\forall R.C &\equiv \exists R.\neg C, \\
\neg\exists R.C &\equiv \forall R.\neg C.
\end{aligned}
$$

An interpretation satisfies (is a model of) a TBox $\mathcal{TB}$ if it satisfies all contained definitions. To define the semantics of assertions, the interpretation function is extended such that it maps each constant a to an element a^I in the interpretation's domain. We say that an assertion $C(a)$ (respectively $R(a, b)$) is *satisfied* by an interpretation I if $a^I \in C^I$ (respectively $(a^I, b^I) \in R^I$). An interpretation I satisfies (is a model of) an ABox $\mathcal{AB}$ if it satisfies all contained assertions. We say that I satisfies (is a model of) a knowledge base $(\mathcal{TB}, \mathcal{AB})$ if I satisfies $\mathcal{TB}$ and $\mathcal{AB}$.

First-order logic semantics of $\mathcal{ALC}$

The semantics of Description Logic is related to first-order logic. Actually, $\mathcal{ALC}$ is a fragment of first-order logic. In other words, since an interpretation I assigns to every atomic concept a unary relation over its domain $\mathcal{D}$ and to every role a binary relation over $\mathcal{D}$, an atomic concept is nothing else than a unary predicate and an atomic role is nothing else than a binary predicate. Consequently the semantics of $\mathcal{ALC}$ can be defined by the semantics of first-order logic and a mapping that assigns to each $\mathcal{ALC}$ expression a formula of first-order logic. This proceeding becomes important for us in Chapter 4.1 because we define there the semantics of $\mathcal{ALC}4$ by a mapping to the four-valued first-order logic introduced in Chapter 3. Thus, we give an example of such a mapping consisting of two[1] functions ϕ in Figure 2.1. The first four lines of Figure 2.1 show the first function $\phi(z)$ that maps assertions and terminological axioms to classical first-order logic. The remaining lines of Figure 2.1 show the second function $\phi(z, x)$ that translates recursively concepts to formulas of first-order logic.

Inference Tasks

The purpose of a KRR system is to perform reasoning tasks on the knowledge of its knowledge base. An $\mathcal{ALC}$ knowledge base $\mathcal{KB}$ has a semantics that corresponds to a set of first-order logic formulas. Thus, inference mechanisms can be applied to make implicit knowledge explicit. The key inference tasks on concepts are defined with respect to a TBox $\mathcal{TB}$ as follows:

---

[1] The first function takes one argument and the second function takes two arguments.

$$\phi(C(a)) \quad \overset{def}{=} \quad \phi(C, a)$$

$$\phi(R(a, b)) \quad \overset{def}{=} \quad R(a, b)$$

$$\phi(C \sqsubseteq D) \quad \overset{def}{=} \quad (\forall x)(\phi(C, x) \rightarrow \phi(D, x))$$

$$\phi(C \equiv D) \quad \overset{def}{=} \quad (\forall x)(\phi(C, x) \leftrightarrow \phi(D, x))$$

$$\phi(\top, x) \quad \overset{def}{=} \quad \top$$

$$\phi(\bot, x) \quad \overset{def}{=} \quad \bot$$

$$\phi(A, x) \quad \overset{def}{=} \quad A(x)$$

$$\phi(\neg C, x) \quad \overset{def}{=} \quad \neg \phi(C, x)$$

$$\phi((C \sqcap D), x) \quad \overset{def}{=} \quad (\phi(C, x) \wedge \phi(D, x))$$

$$\phi((C \sqcup D), x) \quad \overset{def}{=} \quad (\phi(C, x) \vee \phi(D, x))$$

$$\phi(\forall R.C, x) \quad \overset{def}{=} \quad (\forall y)(R(x, y) \rightarrow \phi(C, y)) \text{ with } y \text{ as a new variable}$$

$$\phi(\exists R.C, x) \quad \overset{def}{=} \quad (\exists y)(R(x, y) \wedge \phi(C, y)) \text{ with } y \text{ as a new variable}$$

Figure 2.1: Semantic mapping of $\mathcal{ALC}$ to classical first-order logic.

Satisfiability: A concept C is *satisfiable* with respect to $\mathcal{TB}$ if there exists a model I of $\mathcal{TB}$ such that C^I is not empty. In that case I is said to be a *model* of C.

Subsumption: A concept D *subsumes* a concept C with respect to $\mathcal{TB}$ (written as $\mathcal{TB} \models C \sqsubseteq D$) if $C^I \subseteq D^I$ for every model I of $\mathcal{TB}$.

Equivalence: Two concepts C and D are *equivalent* with respect to $\mathcal{TB}$ (written as $\mathcal{TB} \models C \equiv D$) if $C^I = D^I$ for every model I of $\mathcal{TB}$.

Disjointness: Two concepts C and D are *disjoint* with respect to $\mathcal{TB}$ (written as $\mathcal{TB} \models (C \sqcap D) \equiv \bot$) if $C^I \cap D^I = \emptyset$ for every model I of $\mathcal{TB}$.

The above inference tasks can be expressed exclusively by subsumption as follows:

1. A concept C is *unsatisfiable* if and only if $C \sqsubseteq \bot$.

2. Two concepts C and D are *equivalent* if and only if $C \sqsubseteq D$ and $D \sqsubseteq C$.

3. Two concepts C and D are *disjoint* if and only if $(C \sqcap D) \sqsubseteq \bot$.

The inference task linked to assertions are queries about the relationships between concepts, roles and individuals. The most important tasks with respect to an ABox $\mathcal{AB}$ are defined as follows:

Instance check: An assertion $C(a)$ (respectively $R(a,b)$) is *entailed* by $\mathcal{AB}$ if every interpretation that satisfies $\mathcal{AB}$ also satisfies $C(a)$ (respectively $R(a,b)$). We write $\mathcal{AB} \models C(a)$ (respectively $\mathcal{AB} \models R(a,b)$).

Retrieval problem: The task is to find all individuals a such that $\mathcal{AB} \models C(a)$.

Realisation problem: Given an individual a and a set of concepts, the task is to find the *most specific concept* C from the set such that $\mathcal{AB} \models C(a)$, where C is most specific if it is the most minimal with respect to the subsumption ordering $\sqsubseteq$.

Syntax of $\mathcal{SHIQ}$

$\mathcal{SHIQ}$ is obtained from $\mathcal{ALC}$ by adding the three additional constructors:

1. *role hierarchy* (denoted by $R_1 \sqsubseteq R_2$),

2. *inverse role* (denoted by R^-), and

3. *qualified number restriction* (denoted by $\geq n\, S.C$ and $\leq n\, S.C$), where n is an integer greater or equal than 1 and S is a simple role as defined below.

Let $\mathcal{A_R}$ be the set of $\mathcal{SHIQ}$ role names. The set of roles in $\mathcal{SHIQ}$ is $\mathcal{A_R} \cup \{R^- | R \in \mathcal{A_R}\}$. A $\mathcal{SHIQ}$ *RBox* $\mathcal{RB}$ over the set $\mathcal{A_R}$ is a finite set of transitivity axioms $Trans(R)$ and role inclusion axioms $R_1 \sqsubseteq R_2$, such that $R_1 \sqsubseteq R_2 \in \mathcal{RB}$ iff $R_1^- \sqsubseteq R_2^- \in \mathcal{RB}$, and $Trans(R) \in \mathcal{RB}$ iff $Trans(R^-) \in \mathcal{RB}$. We denote with

$\sqsubseteq^*$ the reflexive-transitive closure of $\sqsubseteq$ and we define a role S to be *simple* if there is no role R such that $R \sqsubseteq^* S$ and R is transitive; otherwise we call S *complex.*

We allow the special infix written binary relation symbols $\approx$ and $\not\approx$ to express equality and inequality of individuals. We extend the notion of an assertion such that it comprises expressions of the form $a \approx b$ and $a \not\approx b$ as well. Consequently, a $\mathcal{SHIQ}$ ABox may contain assertions about the equality and inequality of individuals.

The notion of a concept in $\mathcal{SHIQ}$ is defined as follows:

1. $\top$ and $\bot$ are concepts;

2. each atomic concept A is a concept;

3. if C and D are concepts, and R is a role, then $\neg C$, $(C \sqcap D)$, $(C \sqcup D)$, $\exists R.C$, and $\forall R.C$ are concepts;

4. if C is a concept, S is a simple role, and n is a positive integer, then $\geq n\, S.C$ and $\leq n\, S.C$ are concepts.

A TBox in $\mathcal{SHIQ}$ may comprise not only definitions but also terminological axioms $C \sqsubseteq D$. A $\mathcal{SHIQ}$ *knowledge base* is a triple $(\mathcal{RB}, \mathcal{TB}, \mathcal{AB})$, where $\mathcal{RB}$ is a RBox, $\mathcal{TB}$ is a TBox, and $\mathcal{AB}$ is an ABox.

Semantics of $\mathcal{SHIQ}$

We give the semantics to $\mathcal{SHIQ}$ by extending the mapping ϕ that we introduced above for $\mathcal{ALC}$. Concerning concepts, we define the semantics of qualified number restriction as follows:

$$\phi(\leq n\, S.C, x) \quad \stackrel{def}{=} \quad (\forall y_1, ..., y_{n+1})\big(\bigwedge_{i=1}^{n+1} (S(x, y_i) \wedge \phi(C, y_i)) \rightarrow \bigvee_{i \neq j} y_i \approx y_j \big),$$

$$\phi(\geq n\, S.C, x) \quad \stackrel{def}{=} \quad (\exists y_1, ..., y_n)\big(\bigwedge_{i=1}^{n} (S(x, y_i) \wedge \phi(C, y_i)) \wedge \bigwedge_{i \neq j} y_i \not\approx y_j \big),$$

where n is an integer greater or equal than 1, $\phi(C, y_i)$ is an arbitrary formula with occurrences of a variable $y_i \in \{y_1, ..., y_n\}$, $\bigwedge_{i=1}^{n+1}(S(x, y_i) \wedge \phi(C, y_i))$ stands for the conjunction of $n + 1$ formulas $(S(x, y_i) \wedge \phi(C, y_i))$, and $\bigvee_{i \neq j} y_i \approx y_j$ stands for the disjunction of all formulas $y_i \approx y_j$ with $i \neq j$. The second line has to be read similarly.

The semantics of assertions and RBox axioms is extended as follows:

$$\phi(R) \quad \stackrel{def}{=} \quad (\forall x, y)(R(x, y) \leftrightarrow R^-(y, x)),$$

$$\phi(R_1 \sqsubseteq R_2) \quad \stackrel{def}{=} \quad (\forall x, y)(R_1(x, y) \rightarrow R_2(x, y)),$$

$$\phi(Trans(R)) \quad \stackrel{def}{=} \quad (\forall x, y, z)(R(x, y) \wedge R(y, z) \rightarrow R(x, z)),$$

$$\phi(a \approx b) \quad \stackrel{def}{=} \quad a \approx b,$$

$$\phi(a \not\approx b) \quad \stackrel{def}{=} \quad \neg\, a \approx b.$$

The equivalences of $\mathcal{ALC}$ concepts hold for $\mathcal{SHIQ}$ as well. Furthermore, the following equivalences are valid in $\mathcal{SHIQ}$:

$$\neg \geq n\, S.C \quad \equiv \quad \begin{cases} \forall S.\neg C & \text{if } n = 1, \\ \leq (n-1)S.C & \text{otherwise;} \end{cases}$$
$$\neg \leq n\, S.C \quad \equiv \quad \geq (n+1)S.C.$$

2.3 Paraconsistency by Four-valued Semantics

Informally, we introduced the idea of paraconsistency in Chapter 1. In this section we provide the formal definitions and we illustrate the idea of paraconsistency in a four-valued semantics. Above, we said that the idea of paraconsistency is rooted in the desire to perform reasoning on inconsistent information but so far we did not define what "inconsistent" and "paraconsistent" formally means for a logic. Thus, we give a formal definition of the crucial terms in the following.

Definition 2.3.1
Let $\mathcal{F}$ be a set of formulas and F a formula of the language $\mathcal{L}$ based on the alphabet $\mathcal{A}$. We define with respect to a logic $\langle \mathcal{A}, \mathcal{L}, \models \rangle$:

1. $\mathcal{F}$ is *inconsistent* iff there exists an F such that $\mathcal{F} \models F$ and $\mathcal{F} \models \neg F$.

2. $\mathcal{F}$ is *trivial* iff it has no models, otherwise $\mathcal{F}$ is *non-trivial*.

    A logic that admits trivial sets of formulas is called *trivialisable*.

3. $\mathcal{F}$ is *paraconsistent* iff it is inconsistent but non-trivial.

    A logic is *paraconsistent* iff it allows paraconsistent sets of formulas and it is *fully paraconsistent* iff all subsets of $\mathcal{L}$ are non-trivial.

We use the set of the first-order logic truth values t and f as basis to express four truth values. The four truth values are the elements of the power set of $\{t, f\}$, namely $\{\}$, $\{f\}$, $\{t\}$, and $\{t, f\}$. The syntax of these four truth values reflects intuitively the intended meaning of *lack of knowledge*, *falsehood*, *truth* and *contradiction*, respectively. The classical truth values become independent from each other. This means for a formula F that it can be t and f at the same time if the truth value $\{t, f\}$ is assigned to it.

Example 2.3.2
We consider the set $\{F, \neg F\}$ for which we find $\{F, \neg F\} \models F$ and $\{F, \neg F\} \models \neg F$. Consequently, $\{F, \neg F\}$ is inconsistent. To avoid that $\{F, \neg F\}$ be trivial, we require an interpretation I and a state σ that constitute a model of $\{F, \neg F\}$. Thus, I, σ has to satisfy both F and $\neg F$ simultaneously, which in turn is the case if we find that $I, \sigma \models F$ and $I, \sigma \models \neg F$. For a four-valued logic, we redefine the relation $\models$ such that $I, \sigma \models F$ if and only if $t \in F^{I,\sigma}$. That is to say, F is satisfied if I assigns the truth value $\{t\}$ or the truth value $\{t, f\}$ with respect to

σ. In the case that I assigns $\{t, f\}$ to F, the truth value of $\neg F$ is $\{t, f\}$ as well and, thus, I, σ satisfies F and $\neg F$ simultaneously and I, σ is a model of $\{F, \neg F\}$.

In literature, a special term is used to denote the truth values for which a formula is said to be satisfiable, namely *designated* truth values [PT04, Fit95, AA96, dCBB95]. We use the same term in this thesis for the truth values $\{t\}$ and $\{t, f\}$. The remaining truth values $\{f\}$ and $\{\}$ are said to be *non-designated* truth values.

In [Bel76], Belnap introduces two orderings of the four truth values. The first ordering refers to the *amount of knowledge about truth* symbolised by the truth values and the second ordering refers to the *amount of truth* symbolised by the truth values. In this thesis, we denote the first ordering by $\leq_k$ and the second ordering by $\leq_t$. To give an example, we find for the truth values $\{t\}$ and $\{t, f\}$ that $\{t\} \leq_k \{t, f\}$ and $\{t, f\} \leq_t \{t\}$. Furthermore, Belnap defines the lattice $(B, \leq_k)$ as *approximation lattice* and the lattice $(B, \leq_t)$ as *logical lattice*, where $B = \{\{\}, \{f\}, \{t\}, \{t, f\}\}$. In this thesis, we use these terms as well. In [Gin88], Ginsberg introduced the notion of *bilattices* and he defines in [Gin90, Page 274]: "Essentially, a bilattice is a set equipped with two partial orders and a negation operation that inverts one of them while leaving the other unchanged:

Definition 2.1   *A bilattice is a sextuple $(B, \wedge, \vee, \cdot, +, \neg)$ such that:*

1. *$(B, \wedge, \vee)$ and $(B, \cdot, +)$ are both complete lattices, and*

2. *$\neg : B \to B$ is a mapping with:*

   (a) *$\neg^2 = 1$, and*

   (b) *$\neg$ is a lattice homomorphism from $(B, \wedge, \vee)$ to $(B, \vee, \wedge)$ and from $(B, \cdot, +)$ to itself.*

A bilattice will be called distributive *if the bilattice operations $\wedge, \vee, \cdot$ and $+$ [are][2] distributive with respect to one another."*

Bilattices were further investigated and applied for logic programming by Fitting [Fit94, Fit91b, Fit91a, Fit90]. In [AA96], the idea of a bilattice is advanced by Arieli and Avron to a so-called *logical bilattice*. Arieli and Avron refer to the approximation lattice and the logical lattice of Belnap, and they define as logical bilattice the structure $(B, \leq_t, \leq_k, \neg)$ illustrated in Figure 2.2. In [AA96], it is shown that $(B, \leq_t, \leq_k, \neg)$ is a distributive bilattice according to the Definition 2.1 of Ginsberg. The corresponding lattice operations[3] are the meet and the join operations on the lattices $(B, \leq_t)$ and $(B, \leq_k)$. With respect to $(B, \leq_t)$, the meet operation is denoted by $\wedge$ and the join operation is denoted by $\vee$. The operations $\wedge$ and $\vee$ are associated with the idea of *conjunction* and *disjunction* in a logic. Let F and G be two formulas, then a designated truth value is assigned to $F \wedge G$ if and only if both F and G take designated truth

---

[2]Author's note.

[3]According to Ginsberg's definition $\wedge, \vee, \cdot$ and $+$.
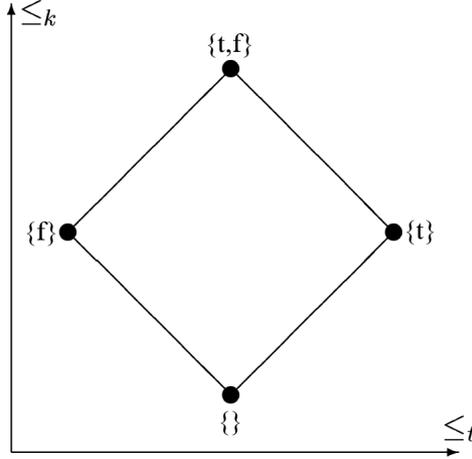
Figure 2.2: Logical bilattice $(B, \leq_t, \leq_k, \neg)$ [AA96].

values. $F \vee G$ takes a designated truth value if and only if F or G take a designated truth value. Corresponding to $(B, \leq_k)$ the meet operation is denoted by $\otimes$ and the join operation is denoted by $\oplus$. The operations $\otimes$ and $\oplus$ are called *consensus* and *gullibility*, respectively. The intuitive meaning of consensus is that $F \otimes G$ reflects the most information that F and G agree on. This means with respect to truth values that $f \in (F \otimes G)^{I,\sigma}$ if and only if $f \in F^{I,\sigma}$ and $f \in G^{I,\sigma}$, and $t \in (F \otimes G)^{I,\sigma}$ if and only if $t \in F^{I,\sigma}$ and $t \in G^{I,\sigma}$. The intuitive meaning of gullibility is that "anything is accepted" [Fit94]. This means with respect to truth values $f \in (F \oplus G)^{I,\sigma}$ if and only if $f \in F^{I,\sigma}$ or $f \in G^{I,\sigma}$, and $t \in (F \oplus G)^{I,\sigma}$ if and only if $t \in F^{I,\sigma}$ or $t \in G^{I,\sigma}$. With respect to negation $\neg$, it is important to notice that:

1. if $F^{I,\sigma} \leq_t G^{I,\sigma}$, then $(\neg G)^{I,\sigma} \leq_t (\neg F)^{I,\sigma}$,

2. if $F^{I,\sigma} \leq_k G^{I,\sigma}$, then $(\neg F)^{I,\sigma} \leq_k (\neg G)^{I,\sigma}$, and

3. $(\neg \neg F)^{I,\sigma} = F^{I,\sigma}$.

The interesting point for paraconsistent reasoning is the fact that one can define a paraconsistent four-valued logic based on a logical bilattice where an intuitive meaning of basic operations is given. A first approach to a four-valued logic is provided by Belnap in [Bel76]. However, Belnap's four-valued logic lacks of an implication connective that can be used within formulas. Concerning this fact it is important to notice that implication in a four-valued logic does not describe logical entailment if it is defined by negation and disjunction as in classical first-order logic. We elucidate this fact with respect to the classical first-order logic $\langle \mathcal{A}, \mathcal{L}, \models \rangle$ introduced in Section 2.1.

Example 2.3.3
In the classical first-order logic $\langle \mathcal{A}, \mathcal{L}, \models \rangle$ the equivalence $\neg F \vee G \equiv F \rightarrow G$

holds (as can be seen with respect to the definition of $\models$ in Section 2.1). Thus, it is possible to express the implication $F \rightarrow F$ by $\neg F \vee F$. Furthermore, we find with respect to the entailment relation $\models$ and the formula F that:

$$\{F\} \models F \quad \text{iff} \quad \text{all models } I, \sigma \text{ of } \{F\} \text{ are models of } F \tag{2.1}$$

$$\text{iff} \quad \text{all models } I, \sigma \text{ of } F \text{ are models of } F \tag{2.2}$$

which in turn means that

$$I, \sigma \not\models F \quad \text{or} \quad I, \sigma \models F. \tag{2.3}$$

In classical logic, we are used to express this as

$$I, \sigma \models \neg F \vee F \tag{2.4}$$

and, thus,

$$I, \sigma \models F \rightarrow F. \tag{2.5}$$

However, (2.3) and (2.4) are not equivalent in Belnap's four-valued logic because in Belnap's four-valued logic $I, \sigma \not\models F$ does not mean that $I, \sigma \models \neg F$. This becomes obvious if we assume that $F^{I,\sigma} = \{\}$ and, thus[4], $(\neg F \vee F)^{I,\sigma} = \{\}$. However, $\{F\} \models F$ holds because all models of $\{F\}$ are models of F. Consequently, the definition of an implication connective as in classical first-order logic does not allow to express entailment properly in Belnap's four-valued logic.

In [AA96, AA98], Arieli and Avron introduce various implication connectives and describe their properties. The use of a certain implication has always to be seen with respect to some properties that are required. In this thesis, we require some special properties that provoke us to analyse thoroughly how an implication connective should be defined and that resulted in the definition of a new implication connective. In Chapter 3.5, we describe this in detail and we compare our implication connective to one of the implication connectives in [AA96, AA98].

---

[4]If $F^{I,\sigma} = \{\}$ then $(\neg F)^{I,\sigma} = \{\}$ because if $F^{I,\sigma} \leq_k \{\}$, then $(\neg F)^{I,\sigma} \leq_k \{\}$.

Chapter 3

Four-valued First-order Logic

This chapter covers the major part of our work. Based on Chapter 2.3, we present the theoretical foundations for paraconsistent reasoning. We develop the required algorithms for translating formulas of our four-valued first-order logic to a representation in two-valued first-order logic that allows paraconsistent reasoning in a classical first-order logic. The relation of four-valued first-order logic to classical first-order logic is stated in the principle theorems (Theorem 3.3.3 and Theorem 3.8.1) of this thesis.

The chapter is structured into eight sections that describe an incremental definition of our four-valued first-order logic. The first three sections cover the basic syntax, semantics and algorithms. In the following sections, we enrich our four-valued first-order logic consecutively by a Truth and a False Formula, an implication connective, an equivalence connective, and an equality predicate. The introduction of the implication connective is of major importance for the entire work. In the eighth section, we lay final foundations for the definition of Description Logics in Chapter 4 by introducing a supplementary connective that is required for $\mathcal{ALC}4$ and $\mathcal{SHIQ}4$.

3.1 Basic Syntax

Basically, formulas of four-valued first-order logic are built up in the same way as the classical first-order logic formulas are. Thus, we have a set of variables $\mathcal{A_V}$, a set of constant symbols $\mathcal{A_C}$, a set of function symbols $\mathcal{A_F}$, a set of relation symbols $\mathcal{A_R}$, the set $\{\forall, \exists\}$ of quantifiers, and a set of punctuation symbols that comprises "(", ")" and ",". These six sets, together with the set $\{\wedge, \vee, \neg\}$ of connectives, constitute the alphabet $\mathcal{A}$ of the basic language $\mathcal{L}$. For notational convenience, we abbreviate the writing of a tuple $(t_1, ..., t_n)$ of terms by a bold letter $\boldsymbol{t}$.

3.2 Basic Semantics

As truth values of our four-valued first-order logic, we consider $\{\}$, $\{f\}$, $\{t\}$, and $\{t, f\}$ as introduced in Chapter 2.3. The syntax of these truth values reflects intuitively the intended meaning of *lack of knowledge*, *falsehood*, *truth*,

and *contradiction*, respectively. In our four-valued first-order logic the classical truth values become independent from each other. This leads to the following extension of the notion of an interpretation. We remember that in classical first-order logic, an interpretation assigns to an n-ary relation symbol R a subset R_+^I of the interpretation's domain $\mathcal{D}$. The elements of R_+^I are exactly those n-tuples of elements of $\mathcal{D}$ for which R is evaluated as *true*. All the other tuples for which R is interpreted as *false* are in the complementary set $R_-^I = \mathcal{D} \backslash R_+^I$. Thus, for each relation symbol, an interpretation partitions the set of n-tuples over its domain into two disjoint sets R_+^I and R_-^I. This changes in four-valued first-order logic. We account for the independence of the truth values $\{t\}$ and $\{f\}$ by allowing the sets R_+^I and R_-^I to overlap and their union to be a proper subset of the interpretation's domain. Consequently, an n-tuple can be in one of four disjoint sets, namely $R_+^I \backslash R_-^I$, $R_-^I \backslash R_+^I$, $R_+^I \cap R_-^I$, and $\mathcal{D} \backslash (R_+^I \cup R_-^I)$. Each set corresponds to one truth value. Formally, we define an interpretation in four-valued first-order logic as follows.

Definition 3.2.1
An *interpretation* I consists of a non-empty domain $\mathcal{D}$ and a mapping ^{I} that assigns

1. to each constant symbol c an element c^I in the domain $\mathcal{D}$;

2. to each n-ary function symbol f an n-ary function $f^I : \mathcal{D}^n \to \mathcal{D}$ over the domain $\mathcal{D}$;

3. to each n-ary relation symbol R an ordered pair $R^I = \langle R_+^I, R_-^I \rangle$ of subsets of $\mathcal{D}^n$.

The assignment of two sets to each relation symbol allows us to link unambiguously one of the four truth values to an instance $R(\boldsymbol{d})$, where $\boldsymbol{d}$ stands for some tuple of ground terms. The truth value $R(\boldsymbol{d})^I$ of $R(\boldsymbol{d})$ depends on the set-membership of $\boldsymbol{d}^I$ according to the following assignments:

1. $R(\boldsymbol{d})^I = \{\}$ iff $\boldsymbol{d}^I \notin R_+^I$ and $\boldsymbol{d}^I \notin R_-^I$,

2. $R(\boldsymbol{d})^I = \{f\}$ iff $\boldsymbol{d}^I \notin R_+^I$ and $\boldsymbol{d}^I \in R_-^I$,

3. $R(\boldsymbol{d})^I = \{t\}$ iff $\boldsymbol{d}^I \in R_+^I$ and $\boldsymbol{d}^I \notin R_-^I$,

4. $R(\boldsymbol{d})^I = \{t, f\}$ iff $\boldsymbol{d}^I \in R_+^I$ and $\boldsymbol{d}^I \in R_-^I$.

The idea is similar to classical first-order logic where we have the limited case that an interpretation assigns either $\{t\}$ or $\{f\}$. Thus, we skip an example.

Definition 3.2.2
A *state* σ over the domain $\mathcal{D}$ of an interpretation is a mapping assigning an element of $\mathcal{D}$ to each variable occurring in $\mathcal{A}_\mathcal{V}$. The meaning of a term t with respect to a state σ and an interpretation I is defined as follows:

$$t^{I,\sigma} \stackrel{def}{=} \begin{cases} \sigma(t) & \text{if } t \text{ is a variable,} \\ c^I & \text{if } t \text{ is a constant } c, \\ f^I(t_1^{I,\sigma}, ..., t_n^{I,\sigma}) & \text{if } t \text{ is of the form } f(t_1, ..., t_n). \end{cases}$$

1. $I, \sigma \models_t R(\boldsymbol{t})$ for an atomic formula $R(\boldsymbol{t})$ iff $\boldsymbol{t}^{I,\sigma} \in R_+^I$,
2. $I, \sigma \models_f R(\boldsymbol{t})$ for an atomic formula $R(\boldsymbol{t})$ iff $\boldsymbol{t}^{I,\sigma} \in R_-^I$,

3. $I, \sigma \models_t \neg F$ iff $I, \sigma \models_f F$,
4. $I, \sigma \models_f \neg F$ iff $I, \sigma \models_t F$,

5. $I, \sigma \models_t F \wedge G$ iff $I, \sigma \models_t F$ and $I, \sigma \models_t G$,
6. $I, \sigma \models_f F \wedge G$ iff $I, \sigma \models_f F$ or $I, \sigma \models_f G$,

7. $I, \sigma \models_t F \vee G$ iff $I, \sigma \models_t F$ or $I, \sigma \models_t G$,
8. $I, \sigma \models_f F \vee G$ iff $I, \sigma \models_f F$ and $I, \sigma \models_f G$,

9. $I, \sigma \models_t (\exists x)F$ iff $I, \sigma\{x \mapsto d\} \models_t F$ for some $d \in \mathcal{D}$,
10. $I, \sigma \models_f (\exists x)F$ iff $I, \sigma\{x \mapsto d\} \models_f F$ for all $d \in \mathcal{D}$,

11. $I, \sigma \models_t (\forall x)F$ iff $I, \sigma\{x \mapsto d\} \models_t F$ for all $d \in \mathcal{D}$,
12. $I, \sigma \models_f (\forall x)F$ iff $I, \sigma\{x \mapsto d\} \models_f F$ for some $d \in \mathcal{D}$.

Figure 3.1: The entailment relations of four-valued first-order logic.

We denote with $\sigma\{x \mapsto d\}$ the state obtained from σ by assigning d to x while leaving all other assignments unchanged.

$$\sigma\{x \mapsto d\}(y) \stackrel{def}{=} \begin{cases} d & \text{if } x = y, \\ \sigma(y) & \text{if } x \neq y. \end{cases}$$

Definition 3.2.2 equals the definition for classical first-order logic given in Chapter 2.1. The only thing that changes is that the interpretation I is different. Thus, we skip an example.

Given an interpretation I and a state σ, we define the notion of truth for all formulas F of four-valued first-order logic by two entailment relations. $\models_t$ expresses the entailment of truth and $\models_f$ expresses the entailment of falsehood. Figure 3.1 shows the definition of $\models_t$ and $\models_f$ which we call in the following *Truth Entailment Relation* and *False Entailment Relation*, respectively.

According to the assignment of truth values to relation symbols, we extend the interpretation's mapping ^{I} such that it covers entire formulas. An interpretation and a state assign a truth value to a formula with respect to $\models_t$ and $\models_f$ as specified by the following scheme:

1. $F^{I,\sigma} = \{\}$ iff $I, \sigma \not\models_t F$ and $I, \sigma \not\models_f F$,

2. $F^{I,\sigma} = \{f\}$ iff $I, \sigma \not\models_t F$ and $I, \sigma \models_f F$,

3. $F^{I,\sigma} = \{t\}$ iff $I, \sigma \models_t F$ and $I, \sigma \not\models_f F$,

4. $F^{I,\sigma} = \{t, f\}$ iff $I, \sigma \models_t F$ and $I, \sigma \models_f F$.

We adhere to the notation introduced in Chapter 2.1 for classical first-order logic by denoting the truth value of a formula F by $F^{I,\sigma}$. Whenever we refer to arbitrary states, we abbreviate $F^{I,\sigma}$ by F^I. Furthermore, we denote our four-valued first-order logic in the style introduced in Chapter 2.1 by $\langle \mathcal{A}, \mathcal{L}, \models_t \rangle$.

A formula F is *satisfiable* iff there exist an interpretation I and a state σ such that $I, \sigma \models_t F$. In this case, we say that I *satisfies* F with respect to σ and that I, σ is a *model* of F. Furthermore, I, σ is a model of a set $\mathcal{F}$ of formulas iff I, σ is a model of all formulas in $\mathcal{F}$. These definitions are similar to classical first-order logic and the only important innovation is that we define satisfiability with respect to the Truth Entailment Relation.

Definition 3.2.3
We call two formulas F and G *semantically equivalent* and we write $F \equiv G$ if we find for all interpretations I and all states σ that:

$$F^{I,\sigma} = G^{I,\sigma}.$$

Definition 3.2.3 is identical to the definition of semantical equivalence in classical first-order logic given in Chapter 2.1. Thus, we skip an example.

The major reasoning task that we want to solve is the question for the logical consequences of a set of formulas $\mathcal{F}$ and especially whether the truth of a certain formula F is entailed by $\mathcal{F}$. In the following, we abbreviate "$\mathcal{F}$ entails the truth of F" by just saying "$\mathcal{F}$ entails F".

Definition 3.2.4
A set of formulas $\mathcal{F}$ *entails* a formula F (written $\mathcal{F} \models_t F$) iff all models of $\mathcal{F}$ are models of F too. Concerning two sets $\mathcal{F}$ and $\mathcal{G}$ of formulas, we say that $\mathcal{F} \models_t \mathcal{G}$ iff:

If I, σ is a model of $\mathcal{F}$, then I, σ is a model of *some* formula of $\mathcal{G}$.

Once again, we find that Definition 3.2.4 is similar to the respective definition in classical first-order logic given in Chapter 2.1. Therefore, we skip an example for Definition 3.2.4 and we prefer to give an example for the advantage of our four-valued first-order logic with respect to inconsistencies.

Example 3.2.5
We formalise the introductory example by defining three different sets $\mathcal{E}_1$, $\mathcal{E}_2$, and $\mathcal{E}_3$ that comprise the information that the German, the Czech, and the historian gave us, respectively:

$$
\begin{aligned}
\mathcal{E}_1 &= \{German(franzKafka),\ Writer(franzKafka)\}, \\
\mathcal{E}_2 &= \{Czech(franzKafka),\ \neg German(franzKafka)\},\ \text{and} \\
\mathcal{E}_3 &= \{AustroHungarian(franzKafka),\ \neg German(franzKafka), \\
&\qquad \neg Czech(franzKafka)\}.
\end{aligned}
$$

We denote the union of the three sets by $\mathcal{E}_4$. Let I_e be an interpretation that assigns the following truth values to the atoms of $\mathcal{E}_4$, where σ is an arbitrary state:

$$
\begin{aligned}
\textit{German(franzKafka)}^{I_e,\sigma} &= \{t, f\}, \\
\textit{Czech(franzKafka)}^{I_e,\sigma} &= \{t, f\}, \\
\textit{AustroHungarian(franzKafka)}^{I_e,\sigma} &= \{t\}, \\
\textit{Writer(franzKafka)}^{I_e,\sigma} &= \{t\} \, .
\end{aligned}
$$

Obviously, $\mathcal{E}_4$ is inconsistent. However, I_e is a model of $\mathcal{E}_4$ and, consequently, $\mathcal{E}_4$ does not explode. We can safely conclude that Franz Kafka is a writer because

$$\mathcal{E}_4 \models_t \textit{Writer(franzKafka)},$$

while not arbitrary formulas are entailed by $\mathcal{E}_4$. The above example illustrates that our four-valued first-order logic is paraconsistent but our concern is to obtain full paraconsistency[1]. By proving the following proposition, we show that our logic is fully paraconsistent as well.

Proposition 3.2.6
The four-valued first-order logic $\langle \mathcal{A}, \mathcal{L}, \models_t \rangle$ is fully paraconsistent.

Proof of Proposition 3.2.6
For the property of full paraconsistency, we have to show that the logic is not trivialisable. Since a set of formulas is trivial if and only if it does not have a model, we show that all sets of formulas have a model. A set of formulas is representable as one single formula, namely the conjunction of all the contained formulas. Hence, it suffices to show that every formula in $\mathcal{L}$ has a model.

We define an interpretation I that assigns to all atoms $R(\boldsymbol{t})$ for all σ and for all $\boldsymbol{t}$ the truth value $\{t, f\}$. That is to say, we find for all σ and for all $\boldsymbol{t}$ that $\boldsymbol{t}^{I,\sigma} \in R_+^I$ and $\boldsymbol{t}^{I,\sigma} \in R_-^I$. We show by induction over the structure of formulas that there exist a model I, σ for every formula in $\mathcal{L}$.

Induction Basis:

1. If F is atomic, then $F^{I,\sigma} = \{t, f\}$ for an arbitrary σ due to the definition of I.

Induction Steps:

2. If F is of the form $\neg F'$ and I, σ is a model of F' such that $F'^{I,\sigma} = \{t, f\}$, then $(\neg F')^{I,\sigma} = \{t, f\}$. Hence, I, σ is a model of F.

3. If F is of the form $G \wedge H$ and I, σ is a model of G and of H such that $G^{I,\sigma} = \{t, f\}$ and $H^{I,\sigma} = \{t, f\}$ then $(G \wedge H)^{I,\sigma} = \{t, f\}$. Hence, I, σ is a model of F.

---

[1]With respect to Definition 2.3.1 in Chapter 2.3.

4. If F is of the form $G \vee H$ and I, σ is a model of G and of H such that $G^{I,\sigma} = \{t, f\}$ and $H^{I,\sigma} = \{t, f\}$ then $(G \vee H)^{I,\sigma} = \{t, f\}$. Hence, I, σ is a model of F.

5. Let F be of the form $(\forall x)F'$. Due to the definition of the interpretation I, we find that I assigns to F' for all variable assignments $\{x \mapsto d\}$ with $d \in \mathcal{D}$ the truth value $\{t, f\}$. Due to the induction hypothesis, we find a state σ such that $F'^{I,\sigma\{x \mapsto d\}} = \{t, f\}$ for all $d \in D$. Hence, I, σ is a model of F.

6. Let F be of the form $(\exists x)F'$. Due to the definition of the interpretation I, we find that I assigns to F' for some variable assignment $\{x \mapsto d\}$ with $d \in \mathcal{D}$ the truth value $\{t, f\}$. Due to the induction hypothesis, we find a state σ such that $F'^{I,\sigma\{x \mapsto d\}} = \{t, f\}$ for all $d \in D$. Hence, I, σ is a model of F.

Since there is some model I, σ of every formula, we find a model of arbitrary sets of formulas. Hence, there are no trivial sets of formulas in our four-valued first-order logic and, consequently, the logic is fully paraconsistent. $\square$

3.3   Basic Algorithms

As we pointed out in the introductory chapter, today's reasoning systems commonly work with classical logics. Thus, our purpose is the calculation of the Truth Entailment Relation by standard reasoning systems for classical first-order logic. To this end, we develop a method for transforming formulas of four-valued first-order logic to formulas of classical first-order logic. The transformation is made in two steps. First, we rewrite formulas of four-valued first-order logic to their *Negation Normal Form*, i.e., a representation where all negation connectives occur in front of atoms. Second, we map the formulas to a semantically equivalent representation in a classical first-order logic. Both steps require some preparatory work.

The transformation of formulas into their Negation Normal Form requires the validity of the Replacement Theorem for formulas of four-valued first-order logic.

Theorem 3.3.1 (Replacement Theorem)
Let G and H be formulas. Let $F[G]$ denote a formula, where G is a subformula, and let $F[G/H]$ denote the formula $F[G]$, where G is replaced by H. $F[G]$ is semantically equivalent to $F[G/H]$ if G and H are semantically equivalent.

Proof of Theorem 3.3.1
We prove the Replacement Theorem by structural induction. We assume that G is semantically equivalent to H.

Induction Basis: F is atomic and so it does not have subformulas other than G. Consequently, $F[G]$ is identical to G and $F[G/H]$ is identical to H. Since

G is semantically equivalent to H, we find that $F[G]$ is semantically equivalent to $F[G/H]$.

Induction Step: If $F = G$, then the claim is trivial. Thus, we assume that $F \neq G$. We have to consider many different formula structures but we find that the proof is similar for all of them. Therefore, we distinguish two different cases of formula structures and we give an example proof for one formula of each case. The proof of other formulas matching the same case is alike.

The first case covers formulas with propositional connectives including negation, conjunction, and disjunction. We show the proof for the conjunction of two formulas. Let $F[G]$ be of the form $F_1[G] \wedge F_2$. Thus, $F[G/H]$ is of the form $F_1[G/H] \wedge F_2$. According to the induction hypothesis, $F_1[G]$ is semantically equivalent to $F_1[G/H]$ and all interpretations I and states σ map $F_1[G]$ to the same truth value as $F_1[G/H]$. By case analysis, we find that $F_1[G] \wedge F_2$ is mapped to the same truth value as $F_1[G/H] \wedge F_2$ and, consequently, $F_1[G] \wedge F_2$ is semantically equivalent to $F_1[G/H] \wedge F_2$. The same argument holds for a subformula G in F_2 such that $F[G]$ is of the form $F_1 \wedge F_2[G]$ and $F[G/H]$ is of the form $F_1 \wedge F_2[G/H]$. Likewise, our argument obviously holds for the case that G occurs in both F_1 and F_2.

The second case covers first-order connectives $\exists$ and $\forall$. We show the proof for an existentially quantified formula. Let $F[G]$ be of the form $(\exists x)F_1[G]$. Thus, $F[G/H]$ is of the form $(\exists x)F_1[G/H]$. With respect to quantified formulas, we distinguish two cases depending on the occurrences of the quantified variable. The case 2.1, where the quantified variable does not occur in G or in H, and the case 2.2, where the quantified variable occurs in both G and in H. For the imaginary case that the quantified variable occurs only in one of the subformulas G or H, we find that G and H are not semantically equivalent and, thus, we do not regard this case.

In the case 2.1 the quantifier does not matter and we find according to the induction hypothesis that $F_1[G]$ is semantically equivalent to $F_1[G/H]$ and, thus, all interpretations I and all states σ map $F_1[G]$ to the same truth value as $F_1[G/H]$. By case analysis, we find that $(\exists x)F_1[G]$ is mapped to the same truth value as $(\exists x)F_1[G/H]$ and, consequently, $(\exists x)F_1[G]$ is semantically equivalent to $(\exists x)F_1[G/H]$.

We consider the case 2.2, where the variable x occurs in G and in H. The induction hypothesis implies that $F_1[G]$ is mapped to the same truth value as $F_1[G/H]$ for all I and for all σ. If we regard all σ, then all variable assignments $\{x \mapsto d\}$ for all $d \in \mathcal{D}$ are considered. This includes an existentially quantified variable x and, therefore, we find that $(\exists x)F_1[G]$ is mapped to the same truth value as $(\exists x)F_1[G/H]$, which in turn means that $F[G]$ is semantically equivalent to $F[G/H]$. $\qquad\square$

We find that the classical equivalences as for instance the laws of de Morgan[2] hold in four-valued first-order logic too. A table of valid equivalences

---

[2]$\neg(F \wedge G) \equiv \neg F \vee \neg G$, and $\neg(F \vee G) \equiv \neg F \wedge \neg G$.

Input: A four-valued first-order logic formula F_4.
Proceeding: Apply the rules below until Negation Normal Form is established. Every rule has two parts, the upper part and the lower part. A rule is applied by replacing an instance of the upper part by an instance of the lower part.
Output: Negation Normal Form of F_4.

$$\frac{\neg\neg F}{F} \qquad \frac{\neg(F\wedge G)}{(\neg F\vee\neg G)} \qquad \frac{\neg(F\vee G)}{(\neg F\wedge\neg G)} \qquad \frac{\neg(\exists x)F}{(\forall x)\neg F} \qquad \frac{\neg(\forall x)F}{(\exists x)\neg F}$$

Figure 3.2: Negation Normal Form Algorithm.

and proofs are given in Appendix A. With respect to those equivalences, the Negation Normal Form Algorithm shown in Figure 3.2 follows immediately.

For the mapping of formulas of four-valued first-order logic to classical first-order logic, we consider a logic $\langle\mathcal{A}_2,\mathcal{L}_2,\models_2\rangle$ without negation connective. We define the alphabet $\mathcal{A}_2$ such that the set of variables, the set of constant symbols, the set of function symbols, the set of quantifiers, and the set of punctuation symbols are identical to the respective sets in $\mathcal{A}$. The set $\mathcal{A}_{\mathcal{R}2}$ of predicate symbols of $\mathcal{A}_2$ differs from the set $\mathcal{A}_{\mathcal{R}}$ of predicate symbols of $\mathcal{A}$. We determine for every predicate symbol $R \in \mathcal{A}_{\mathcal{R}}$ two predicate symbols R_+ and R_- in $\mathcal{A}_{\mathcal{R}2}$. The set of connectives of $\mathcal{A}_2$ comprises $\wedge$ and $\vee$.

Let $\mathcal{F}$ be a set of formulas of $\langle\mathcal{A},\mathcal{L},\models_t\rangle$ in Negation Normal Form. We denote with $\mathcal{F}_\lambda$ the set $\mathcal{F}$ whose formulas are mapped to $\langle\mathcal{A}_2,\mathcal{L}_2,\models_2\rangle$. Figure 3.3 shows the function λ that maps formulas in Negation Normal Form from $\langle\mathcal{A},\mathcal{L},\models_t\rangle$ to $\langle\mathcal{A}_2,\mathcal{L}_2,\models_2\rangle$, and its inverse function λ^{-1} that maps formulas in the opposite direction. To give an example for the mapping, we consider the following case.

Example 3.3.2
Let

$$\begin{aligned} \mathcal{E}_4 \ = \ \{ &German(franzKafka), \ Writer(franzKafka),\\ &Czech(franzKafka), \ AustroHungarian(franzKafka),\\ &\neg German(franzKafka), \ \neg Czech(franzKafka)\} \end{aligned}$$

be a set of formulas of our four-valued first-order logic $\langle\mathcal{A},\mathcal{L},\models_t\rangle$. The mapping of all its formulas to the classical first-order logic $\langle\mathcal{A}_2,\mathcal{L}_2,\models_2\rangle$ yields

$$\begin{aligned} \mathcal{E}_\lambda \ = \ \{ &German_+(franzKafka), \ Writer_+(franzKafka),\\ &Czech_+(franzKafka), \ AustroHungarian_+(franzKafka),\\ &German_-(franzKafka), \ Czech_-(franzKafka)\}. \end{aligned}$$

$$\lambda(F_4) \stackrel{def}{=} \begin{cases} R_+(\boldsymbol{t}) & \text{for positive literals } R(\boldsymbol{t}), \\ R_-(\boldsymbol{t}) & \text{for negative literals } \neg R(\boldsymbol{t}), \\ (\lambda(G) \circ \lambda(H)) & \text{for two formulas } G \text{ and } H \\ & \text{with } \circ \text{ as binary connective,} \\ (Qx)\lambda(F) & \text{for a formula } F \text{ with } Q \text{ as quantifier.} \end{cases}$$

$$\lambda^{-1}(F_2) \stackrel{def}{=} \begin{cases} R(\boldsymbol{t}) & \text{for atoms of the form } R_+(\boldsymbol{t}), \\ \neg R(\boldsymbol{t}) & \text{for atoms of the form } R_-(\boldsymbol{t}), \\ (\lambda^{-1}(G) \circ \lambda^{-1}(H)) & \text{for two formulas } G \text{ and } H \\ & \text{with } \circ \text{ as binary connective,} \\ (Qx)\lambda^{-1}(F) & \text{for a formula } F \text{ with } Q \text{ as quantifier.} \end{cases}$$

Figure 3.3: Functions $\lambda : F_4 \mapsto F_2$ and $\lambda^{-1} : F_2 \mapsto F_4$.

The preparatory steps for paraconsistent reasoning in classical first-order logic are done. It remains to show that we can infer with the classical entailment relation $\models_2$ in $\langle \mathcal{A}_2, \mathcal{L}_2, \models_2 \rangle$ exactly what we can infer with the Truth Entailment Relation $\models_t$ in $\langle \mathcal{A}, \mathcal{L}, \models_t \rangle$. We state this crucial property in the principle theorem of this chapter as follows.

Theorem 3.3.3
The entailment of a set of formulas $\mathcal{G}$ by a set of formulas $\mathcal{F}$ in $\langle \mathcal{A}, \mathcal{L}, \models_t \rangle$ can be calculated by the classical entailment relation $\models_2$ because:

$$\mathcal{F} \models_t \mathcal{G} \quad \text{iff} \quad \mathcal{F}_\lambda \models_2 \mathcal{G}_\lambda.$$

We prove the above theorem by showing that whenever an interpretation I_4 and a state σ constitute a model of $\mathcal{F}$ and a model of some formula of $\mathcal{G}$, then there exists a corresponding interpretation I_2 that constitutes together with σ a model of $\mathcal{F}_\lambda$ and a model of some formula of $\mathcal{G}_\lambda$ and vice versa. To this end, we define a corresponding interpretation I_2 of classical first-order logic for each interpretation I_4 of our four-valued first-order logic as follows.

Definition 3.3.4
Let I_4 be an interpretation in our four-valued first-order logic $\langle \mathcal{A}, \mathcal{L}, \models_t \rangle$ and let $\mathcal{D}$ be its domain, then the domain of the corresponding interpretation I_2 in the classical first-order logic $\langle \mathcal{A}_2, \mathcal{L}_2, \models_2 \rangle$ is identical to $\mathcal{D}$. Let σ be an arbitrary state in $\langle \mathcal{A}, \mathcal{L}, \models_t \rangle$. σ is a state in $\langle \mathcal{A}_2, \mathcal{L}_2, \models_2 \rangle$ as well because the domain of I_4 is identical to the domain of I_2, and the considered set of variables in $\mathcal{A}$ is identical to the considered set of variables in $\mathcal{A}_2$.
We say that I_4 and I_2 *correspond to each other* if:

1. $I_4, \sigma \models_t R(\boldsymbol{t})$ iff $I_2, \sigma \models_2 R_+(\boldsymbol{t})$, and

2. $I_4, \sigma \models_f R(\boldsymbol{t})$ iff $I_2, \sigma \models_2 R_-(\boldsymbol{t})$.

We illustrate this in the following example.

Example 3.3.5
Let I_e be the interpretation and let $\mathcal{E}_4$ be the set of formulas of Example 3.2.5. The corresponding interpretation to I_e is I_2 such that:

$$
\begin{aligned}
German_+(franzKafka)^{I_2,\sigma} &= t, \\
Czech_+(franzKafka)^{I_2,\sigma} &= t, \\
German_-(franzKafka)^{I_2,\sigma} &= t, \\
Czech_-(franzKafka)^{I_2,\sigma} &= t, \\
AustroHungarian_+(franzKafka)^{I_2,\sigma} &= t, \\
Writer_+(franzKafka)^{I_2,\sigma} &= t.
\end{aligned}
$$

For the proof of Theorem 3.3.3, it remains to show that if I_4, σ is a model of $\mathcal{F}$ and a model of some formula of $\mathcal{G}$, then the corresponding interpretation I_2 and σ constitute a model of $\mathcal{F}_\lambda$ and a model of some formula of $\mathcal{G}_\lambda$, and vice versa. I_4, σ is a model of $\mathcal{F}$ if and only if it is a model of all formulas of this set. Thus, we have to show that whenever I_4, σ is a model of a formula $F_4 \in \mathcal{F}$, then I_2, σ is a model of $\lambda(F_4)$ and vice versa. Since we want to prove this for arbitrary sets of formulas, we consider all subsets of $\mathcal{L}$. It suffices to show for all formulas $F_4 \in \mathcal{L}$ that whenever I_4, σ is a model of F_4, then I_2, σ is a model of $\lambda(F_4)$ and vice versa.

Proof of Theorem 3.3.3
We show that $I_4, \sigma \models_t F_4$ iff $I_2, \sigma \models_2 \lambda(F_4)$ for arbitrary states σ.

Only-if: Assume that I_4, σ is a model of a four-valued first-order logic formula F_4. We show by induction on the structure of F_4 that the corresponding interpretation I_2, together with σ, is a model of $\lambda(F_4)$.

Induction Basis:

1. If F_4 is atomic, i.e., of the form $R(\boldsymbol{t})$, then $I_4, \sigma \models_t R(\boldsymbol{t})$. In this case, we find that $I_2, \sigma \models_2 R_+(\boldsymbol{t})$ and, therefore, I_2, σ is a model of $\lambda(F_4)$.

2. If F_4 is of the form $\neg R(\boldsymbol{t})$, then $I_4, \sigma \models_t \neg R(\boldsymbol{t})$. In this case, we find that $I_2, \sigma \models_2 R_-(\boldsymbol{t})$ and, therefore, I_2, σ is a model of $\lambda(F_4)$.

Induction Steps:

3. If F_4 is of the form $G \wedge H$, then $I_4, \sigma \models_t G$ and $I_4, \sigma \models_t H$. The induction hypothesis implies that $I_2, \sigma \models_2 \lambda(G)$ and $I_2, \sigma \models_2 \lambda(H)$. Hence, I_2, σ is a model of $\lambda(F_4)$.

4. If F_4 is of the form $G \lor H$, then $I_4, \sigma \models_t G$ or $I_4, \sigma \models_t H$. The induction hypothesis implies that $I_2, \sigma \models_2 \lambda(G)$ or $I_2, \sigma \models_2 \lambda(H)$. Hence, I_2, σ is a model of $\lambda(F_4)$.

5. If F_4 is of the form $(\forall x)F$, then $I_4, \sigma\{x \mapsto d\} \models_t F$ for all $d \in \mathcal{D}$ with $\mathcal{D}$ as domain of I_4 and of I_2. Due to the induction hypothesis and due to the fact that the domains of I_4 and I_2 are identical, we find that $I_2, \sigma\{x \mapsto d\} \models_2 \lambda(F)$ for all $d \in \mathcal{D}$. Hence, I_2, σ is a model of $\lambda(F_4)$.

6. If F_4 is of the form $(\exists x)F$, then $I_4, \sigma\{x \mapsto d\} \models_t F$ for some $d \in \mathcal{D}$ with $\mathcal{D}$ as domain of I_4 and of I_2. Due to the induction hypothesis and due to the fact that the domains of I_4 and I_2 are identical, we find that $I_2, \sigma\{x \mapsto d\} \models_2 \lambda(F)$ for some $d \in \mathcal{D}$. Hence, I_2, σ is a model of $\lambda(F_4)$.

If: Assume that I_2, σ is a model of a formula F_2 of our classical first-order logic $\langle \mathcal{A}_2, \mathcal{L}_2, \models_2 \rangle$. We show that the corresponding interpretation I_4 and σ constitute a model of $\lambda^{-1}(F_2)$.

Induction Basis:

1. If F_2 is atomic and of the form $R_+(\boldsymbol{t})$, then $I_2, \sigma \models_2 R_+(\boldsymbol{t})$. In this case, we find that $I_4, \sigma \models_t R(\boldsymbol{t})$ and, therefore, I_4, σ is a model of $\lambda^{-1}(F_2)$.

2. If F_2 is atomic and of the form $R_-(\boldsymbol{t})$, then $I_2, \sigma \models_2 R_-(\boldsymbol{t})$. In this case, we find that $I_4, \sigma \models_t \neg R(\boldsymbol{t})$ and, therefore, I_4, σ is a model of $\lambda^{-1}(F_2)$.

Induction Steps[3]:

3. If F_2 is of the form $G \land H$, then $I_2, \sigma \models_2 G$ and $I_2, \sigma \models_2 H$. The induction hypothesis implies that $I_4, \sigma \models_t \lambda^{-1}(G)$ and $I_4, \sigma \models_t \lambda^{-1}(H)$. Hence, I_4, σ is a model of $\lambda^{-1}(F_2)$.

4. If F_2 is of the form $G \lor H$, then $I_2, \sigma \models_2 G$ or $I_2, \sigma \models_2 H$. The induction hypothesis implies that $I_4, \sigma \models_t \lambda^{-1}(G)$ or $I_4, \sigma \models_t \lambda^{-1}(H)$. Hence, I_4, σ is a model of $\lambda^{-1}(F_2)$.

5. If F_2 is of the form $(\forall x)F$, then $I_2, \sigma\{x \mapsto d\} \models_2 F$ for all $d \in \mathcal{D}$ with $\mathcal{D}$ as domain of I_4 and of I_2. Due to the induction hypothesis and due to the fact that the domains of I_4 and I_2 are identical, we find that $I_4, \sigma\{x \mapsto d\} \models_t \lambda^{-1}(F)$ for all $d \in \mathcal{D}$. Hence, I_4, σ is a model of $\lambda^{-1}(F_2)$.

6. If F_2 is of the form $(\exists x)F$, then $I_2, \sigma\{x \mapsto d\} \models_2 F$ for some $d \in \mathcal{D}$ with $\mathcal{D}$ as domain of I_4 and of I_2. Due to the induction hypothesis and due to the fact that the domains of I_4 and I_2 are identical, we find that $I_4, \sigma\{x \mapsto d\} \models_t \lambda^{-1}(F)$ for some $d \in \mathcal{D}$. Hence, I_4, σ is a model of $\lambda^{-1}(F_2)$. □

---

[3]The induction step for negation is omitted since all considered formulas F_2 are negation free.

$$\frac{\neg \top}{\bot} \qquad\qquad \frac{\neg \bot}{\top}.$$

Figure 3.4: Extension of the Negation Normal Form Algorithm for $\top$ and $\bot$.

We conclude the discussion of the basic four-valued first-order logic by a final example.

Example 3.3.6

Let the set $\mathcal{E}_\lambda$ be as in Example 3.3.2. Calculating in classical first-order logic allows us to conclude that Franz Kafka is a writer because

$$\mathcal{E}_\lambda \models_2 \mathit{Writer}_+(\mathit{franzKafka})$$

but it is not possible to conclude arbitrary formulas because $\mathcal{E}_\lambda$ does not explode.

3.4   Four-valued first-order logic with Truth and False Formula

The availability of a Truth and a False Formula is required in the following chapter for the mapping of Description Logics to four-valued first-order logic. Therefore, we introduce the null-ary relation symbols $\top$ and $\bot$ that denote the Truth and the False Formula respectively. Formally, we extend the syntax of our logic by defining the alphabet $\mathcal{A}_\top$ that is like $\mathcal{A}$ but includes $\top$ and $\bot$. According to the alphabet, we denote the language based on $\mathcal{A}_\top$ by $\mathcal{L}_\top$ and we denote the respective logic by $\langle \mathcal{A}_\top, \mathcal{L}_\top, \models_t \rangle$.

We define the semantics of $\top$ and $\bot$ with respect to the entailment relations $\models_t$ and $\models_f$ as follows:

13. $I, \sigma \models_t \top$ for all interpretations I and for all states σ,
14. $I, \sigma \not\models_f \top$ for all interpretations I and for all states σ,

15. $I, \sigma \not\models_t \bot$ for all interpretations I and for all states σ,
16. $I, \sigma \models_f \bot$ for all interpretations I and for all states σ.

The extension of our four-valued first-order logic requires to review the theorems and algorithms that we developed for the basic four-valued first-order logic. Therefore, we extend first the Negation Normal Form Algorithm and the mapping λ, and we rethink afterwards Theorem 3.3.3 and the Replacement Theorem. Finally, we discuss paraconsistency of the extended logic.

The extension of the Negation Normal Form Algorithm is easily done by the rules in Figure 3.4. The extension of the function λ is done in a similarly simple way. The Truth and the False Formula translate one-to-one to the Truth and the False Formula of classical first-order logic, respectively. The semantics of $\top$ and $\bot$ implies that all interpretations I, together with an arbitrary state σ, constitute a model of $\top$ and that no I, σ is a model of $\bot$. Since in classical first-order logic all interpretations I_2, together with an arbitrary state σ, constitute

a model of $\top$ and no I_2, σ is a model of $\bot$, Theorem 3.3.3 is obviously valid for $\langle \mathcal{A}_\top, \mathcal{L}_\top, \models_t \rangle$ too. Further, we find that $\top^I = (\neg \bot)^I$ and $\bot^I = (\neg \top)^I$. That is to say, we can replace occurrences of $\top$ in a formula by $\neg \bot$ and vice versa, and we can replace occurrences of $\bot$ in a formula by $\neg \top$ and vice versa. Neither $\top$ nor $\bot$ are semantically equivalent to any formula of the basic language $\mathcal{L}$. Thus, the above mentioned replacements are all of the new replacements that we have additionally to consider with respect to the Replacement Theorem[4]. Hence, the Replacement Theorem obviously holds for $\langle \mathcal{A}_\top, \mathcal{L}_\top, \models_t \rangle$ as well.

So far, four-valued first-order logic with Truth and False Formula appears to be nice but if we dare a closer look, we encounter a serious problem with respect to paraconsistency. The existence of a Truth or a False Formula gives us the means to create a trivial set of formulas, namely $\{\bot\}$. Hence, we lose full paraconsistency. Regarding our purpose to join different sets of formulas, there remains only the option to check the respective sets for occurrences of $\top$ and $\bot$ connectives. Whenever one of those connectives is found in a formula it must be analysed whether adding the respective formula to the set union trivialises the set union or not.

3.5 Four-valued first-order logic with Implication

With the current basic language of four-valued first-order logic we are able to write simple structured formulas but we cannot formalize a sentence like:

"If a writer has the German nationality, then she is a German writer."

The formula $\neg (Writer(x) \wedge German(x)) \vee GermanWriter(x)$ does not express the intended implication appropriately as we explained in Chapter 2.3. For us, it is important that sentences as the above one, can be expressed in Description Logics by subsumption which in turn translates to first-order logic as implication. Our aspired use of paraconsistent logic is to model expressions of Description Logics. Thus, we require the extension of our four-valued first-order logic by an implication connective. Both the definition of an implication connective and the extension of the algorithms developed so far require a detailed discussion. Therefore, we split this section into four subsections. The first describes the definition of our implication connective that differs from all others that we found in literature. The second subsection covers the extension of the Negation Normal Form Algorithms which includes a detailed discussion of the obstacles that we encountered. The third subsection describes the extension of the mapping λ and in the fourth subsection, we compare our implication connective to the implication $\supset$ in [AA96, AA98].

Defining an Implication Connective

An important property of an implication connective is that it allows to express consequence relations among formulas by other formulas which in turn is given if and only if the Deduction Theorem holds.

---

[4]Cases like $F \wedge \bot$ can be trivially replaced by $\bot$.

Deduction Theorem

$\mathcal{F} \cup \{F\} \models G$ iff $\mathcal{F} \models F \oslash G$, where $\oslash$ denotes an implication connective, $\models$ a logical consequence relation, $\mathcal{F}$ is a set of formulas and F and G are formulas.

Another important property is linked to the representation of formulas in classical first-order logic. Our mapping of formulas from four-valued first-order logic to classical first-order logic works on literals and, thus, we have to transform all formulas into their Negation Normal Form, including those with occurrences of the implication connective.

The definition of an appropriate implication connective demands a thorough discussion of the properties that this connective must satisfy. For our purpose, we require the following:

1. full paraconsistency has to be preserved,

2. the Deduction Theorem has to hold with respect to $\models_t$,

3. formulas with implication have to be reducible to Negation Normal Form.

In the following, we discuss each of these properties in turn and explain the consequences for the definition of an implication connective. We illustrate the consequences by creating a truth table for a general implication connective that we denote by $\oslash$. Figure 3.5 illustrates the truth table of $\oslash$ that satisfies the first two properties.

The first item is obviously necessary to avoid that our logic becomes trivialisable. Trivialisability occurs whenever we can write a formula that does not have a model. Using negation, conjunction, and disjunction, it is not possible to write such a formula because if an interpretation assigns $\{t, f\}$ to two formulas F and G, any newly created formula ($\neg F$, $F \wedge G$, $F \vee G$, ...) is interpreted as $\{t, f\}$ too. We require the same behaviour of an implication connective and contribute to this by defining $\{t, f\} \oslash \{t, f\} = \{t, f\}$. Thus, the lower right field of the truth table in Figure 3.5 is determined to be $\{t, f\}$.

The Deduction Theorem fixes many fields of the truth table. Examining the only-if-direction, we find under assumption of $\mathcal{F} \cup \{F\} \models_t G$ that in order for $\mathcal{F} \models_t F \oslash G$ to hold, all models of $\mathcal{F}$ have to be models of $F \oslash G$ as well. We encounter two cases. The first, if I, σ is a model of F, then it is a model of $\mathcal{F} \cup \{F\}$ and by the assumption it is a model of G too. Because I, σ is a model of $\mathcal{F}$ it has to be a model of $F \oslash G$ as well. That is to say, we have to define $\oslash$ such that $F \oslash G$ takes a designated truth value whenever F and G take designated truth values. The second case, if I, σ is not a model of F, then I, σ has to be a model of $F \oslash G$ since I, σ is a model of $\mathcal{F}$. Consequently, we have to define $\oslash$ such that whenever F does not take a designated truth value the implication $F \oslash G$ is interpreted as either $\{t\}$ or $\{t, f\}$.

We turn to the examination of the if-direction of the Deduction Theorem by assuming that $\mathcal{F} \models_t F \oslash G$ holds. In order for $\mathcal{F} \cup \{F\} \models_t G$ to hold, all models of $\mathcal{F} \cup \{F\}$ have to be models of G as well. Let I, σ be a model of $\mathcal{F} \cup \{F\}$, then it is a model of $F \oslash G$ by the assumption. In order for I, σ

$\oslash$	$\{\}$	$\{f\}$	$\{t\}$	$\{t, f\}$
$\{\}$	$\{t\}$ or $\{t, f\}$	$\{t\}$ or $\{t, f\}$	$\{t\}$ or $\{t, f\}$	$\{t\}$ or $\{t, f\}$
$\{f\}$	$\{t\}$ or $\{t, f\}$	$\{t\}$ or $\{t, f\}$	$\{t\}$ or $\{t, f\}$	$\{t\}$ or $\{t, f\}$
$\{t\}$	$\{\}$ or $\{f\}$	$\{\}$ or $\{f\}$	$\{t\}$ or $\{t, f\}$	$\{t\}$ or $\{t, f\}$
$\{t, f\}$	$\{\}$ or $\{f\}$	$\{\}$ or $\{f\}$	$\{t\}$ or $\{t, f\}$	$\{t, f\}$

Figure 3.5: Truth table of a connective that satisfies the Deduction Theorem and that preserves full paraconsistency.

to be a model of $\mathcal{F} \cup \{F\}$, I, σ has to be a model of F. The crucial point for the Deduction Theorem to hold is the following inference. If I, σ is a model of F and of $F \oslash G$, then I, σ has to be a model of G. This means (under the assumption that F takes a designated truth value) that the truth table of $\oslash$ must assign designated truth values to $F \oslash G$ if and only if G takes a designated truth value. Consequently, if G takes non-designated truth values, $F \oslash G$ must take non-designated truth values.

In short, the values of the third and fourth line of the first and second column in Figure 3.5 have to be either $\{f\}$ or $\{\}$ and the remaining values have to be either $\{t\}$ or $\{t, f\}$.

The third property that we require of an implication connective is the reducibility of formulas with implication to a Negation Normal Form. An intuitive idea to obtain this is representing the implication connective by negation, conjunction, and disjunction because we know how formulas with these connectives are handled. However, this does not work. We run into a dilemma with the second property since in four-valued first-order logic it is impossible to define a connective by negation, conjunction, and disjunction that satisfies the Deduction Theorem. The reason for this is that negation, conjunction, and disjunction are monotonic operations with respect to $\leq_k$ in the approximation lattice $(B, \leq_k)$ (Page 20) [Avr99], where monotonicity is defined as follows.

Definition 3.5.2
Let $\langle \mathcal{B}, \leq \rangle$ be a partially ordered set. An operation $f : \mathcal{B}^n \to \mathcal{B}$ is called *monotonic* (with respect to $\leq$) if $f(\boldsymbol{x}) \leq f(\boldsymbol{y})$ whenever $\boldsymbol{x} \leq \boldsymbol{y}$ (we say that $\boldsymbol{x} = (x_1, ..., x_n) \leq \boldsymbol{y} = (y_1, ..., y_n)$ iff we find for all $1 \leq i \leq n$ that $x_i \leq y_i$).

Obviously, defining an operation f by applying one monotonic operation to another monotonic operation causes f to be monotonic. Now, it is essential to realise that any connective that satisfies the Deduction Theorem has to be a non-monotonic operation with respect to $\leq_k$. Above, we explained the requirements on the truth table of a connective that satisfies the Deduction Theorem and we refer to this truth table in the following. Let F_1, F_2, G_1 and G_2 be formulas to which an interpretation assigns the truth values $\{f\}$, $\{t, f\}$, $\{f\}$ and $\{f\}$, respectively. We find that $F_1^{I,\sigma} \leq_k F_2^{I,\sigma}$ and $G_1^{I,\sigma} \leq_k G_2^{I,\sigma}$. Monotonicity requires that $(F_1 \oslash G_1)^{I,\sigma} \leq_k (F_2 \oslash G_2)^{I,\sigma}$ but regarding the truth table of

$\rightarrow$	$\{\}$	$\{f\}$	$\{t\}$	$\{t,f\}$
$\{\}$	$\{t\}$	$\{t\}$	$\{t\}$	$\{t\}$
$\{f\}$	$\{t\}$	$\{t,f\}$	$\{t\}$	$\{t,f\}$
$\{t\}$	$\{\}$	$\{\}$	$\{t\}$	$\{t\}$
$\{t,f\}$	$\{\}$	$\{f\}$	$\{t\}$	$\{t,f\}$

Figure 3.6: Truth table of the implication connective $\rightarrow$.

Figure 3.5, we notice that this requirement is not satisfiable[5]. Hence, we cannot define our implication by the operations available so far.

A loophole is the definition of an additional negation-like connective $\sim$ that is non-monotonic. We call $\sim$ the *convolution*-connective and we define it by the following truth table with respect to a formula F.

F	$\{\}$	$\{f\}$	$\{t\}$	$\{t,f\}$
$\sim F$	$\{t\}$	$\{t,f\}$	$\{\}$	$\{f\}$

The intuition behind this definition is that both designated truth-values are mapped to non-designated truth-values and vice versa. Of course, this can be achieved by other unary connectives too [Rue96]. However, our purpose is the definition of an implication connective and $\sim$ supports such a definition in an intuitive way as follows:

$$F \rightarrow G \stackrel{def}{=} \ \sim F \vee G.$$

The corresponding truth table is given in Figure 3.6. We observe immediately that (with respect to $\models_t$) the Deduction Theorem holds and that full paraconsistency is preserved. We give the proof for the Deduction Theorem in Appendix A on Page 94. The property of full paraconsistency is considered below in Section 3.8 in Proposition 3.8.2 and we prove this proposition on Page 62. Another useful property of our implication connective is that we obtain a well-known deduction rule for our four-valued first-order logic, namely the *Modus Ponens*:

$$\{F, \ (F \rightarrow G)\} \models_t G.$$

Proposition 3.5.3
The Modus Ponens deduction rule holds in our four-valued first-order logic.

Proof of Proposition 3.5.3
Let I, σ be a model of $\{F, \ (F \rightarrow G)\}$, then we find that $I, \sigma \models_t F \wedge (F \rightarrow G)$, which in turn is the case if and only if $I, \sigma \models_t F$ and in addition $I, \sigma \not\models_t F$ or $I, \sigma \models_t G$. This implies that $I, \sigma \models_t G$ holds and, hence, $\{F, \ (F \rightarrow G)\} \models_t G$ holds. $\square$

---

[5]It is important to notice that $\{f\}$ and $\{t\}$ are not comparable with respect to $\leq_k$.

The drawback of the convolution connective is that it trivialises our logic. To see this, we put the convolution connective in the context of the entailment relations:

$$17. \quad I, \sigma \models_t \sim F \quad \text{iff} \quad I, \sigma \not\models_t F,$$
$$18. \quad I, \sigma \models_f \sim F \quad \text{iff} \quad I, \sigma \models_f F.$$

Example 3.5.4
The formula $F \wedge \sim F$ is not satisfiable since $I, \sigma \models_t F \wedge \sim F$ if and only if $I, \sigma \models_t F$ and $I, \sigma \not\models_t F$. Further, we find that $I, \sigma \models_f F \wedge \sim F$ if and only if $I, \sigma \models_f F$. Consequently, the formula $F \wedge \sim F$ is either $\{f\}$ or $\{\}$ with respect to all four-valued interpretation.

Thus, we do not allow the use of convolution as common connective for writing formulas in four-valued first-order logic. Formally, we extend the syntax of the logic $\langle \mathcal{A}_\top, \mathcal{L}_\top, \models_t \rangle$ by defining the alphabet $\mathcal{A}_\to$ that is like $\mathcal{A}_\top$ but includes $\to$. According to the alphabet, we denote the language based on $\mathcal{A}_\to$ by $\mathcal{L}_\to$ and we denote the respective logic by $\langle \mathcal{A}_\to, \mathcal{L}_\to, \models_t \rangle$.

Extension of the Negation Normal Form Algorithm

For the extension of the Negation Normal Form Algorithm, we encounter two options. The first option considers the use of the convolution connective because we want to transform a formula $\neg(F \to G)$ to a semantically equivalent Negation Normal Form, namely $\neg \sim F \wedge \neg G$. The second option is based on the idea not to consider semantically equivalent transformations for establishing a Negation Normal Form. This is a serious proposal since our goal is the representation of formulas in classical first-order logic where only one of our entailment relations is present, namely the Truth Entailment Relation which is described by the classical entailment relation. With respect to two formulas F and G, we defined semantical equivalence as equivalence of truth values, which in turn means that F and G are semantically equivalent if and only if:

1. $I, \sigma \models_t F$ iff $I, \sigma \models_t G$, and

2. $I, \sigma \models_f F$ iff $I, \sigma \models_f G$.

If we consider exclusively the Truth Entailment Relation in classical first-order logic, we might not care about semantical equivalence. For the following, we extend the notion of equivalence.

Definition 3.5.5
Two formulas F and G are said to be *equivalent with respect to Truth Entailment* if:
$$I, \sigma \models_t F \text{ iff } I, \sigma \models_t G.$$

In this case, we write $F \equiv_t G$. Thus, it may occur that $F \equiv_t G$ but $F \not\equiv G$.

Example 3.5.6
We find for the formula $\neg(F \to G)$ that

1. $\neg(F \to G) \not\equiv \neg F \wedge \neg G$, but

2. $\neg(F \to G) \equiv_t \neg F \wedge \neg G$.

We verify this easily with respect to the Truth Entailment Relation and the False Entailment Relation as follows:

$$
\begin{aligned}
& I, \sigma \models_t \neg(F \to G) \\
\text{iff} \quad & I, \sigma \models_f F \to G \\
\text{iff} \quad & I, \sigma \models_f \sim F \vee G \\
\text{iff} \quad & I, \sigma \models_f \sim F \ \text{ and } \ I, \sigma \models_f G \\
\text{iff} \quad & I, \sigma \models_f F \ \text{ and } \ I, \sigma \models_f G \\
\text{iff} \quad & I, \sigma \models_t \neg F \ \text{ and } \ I, \sigma \models_t \neg G \\
\text{iff} \quad & I, \sigma \models_t \neg F \wedge \neg G,
\end{aligned}
$$

and

$$
\begin{aligned}
& I, \sigma \models_f \neg(F \to G) \\
\text{iff} \quad & I, \sigma \models_t F \to G \\
\text{iff} \quad & I, \sigma \models_t \sim F \vee G \\
\text{iff} \quad & I, \sigma \models_t \sim F \ \text{ or } \ I, \sigma \models_t G \\
\text{iff} \quad & I, \sigma \not\models_t F \ \text{ or } \ I, \sigma \models_t G \\
\text{iff} \quad & I, \sigma \not\models_f \neg F \ \text{ or } \ I, \sigma \models_f \neg G \\
\text{not equivalent to} \quad & I, \sigma \models_f \neg F \wedge \neg G.
\end{aligned}
$$

The above example illustrates the second option to transform the formula $\neg(F \to G)$ to a Negation Normal Form that is equivalent with respect to Truth Entailment. If we assume F and G to be atoms, then $\neg F \wedge \neg G$ is a Negation Normal Form of $\neg(F \to G)$.

In the following, we discuss both options. First, we investigate the option to transform $\neg(F \to G)$ to a semantically equivalent Negation Normal Form by using the convolution connective and, second, we investigate how a Negation Normal Form Algorithm works without convolution connective. In the remainder of this work, we consider the second Negation Normal Form Algorithm without convolution connective. Thus, the reader that is not interested in the discussion of different Negation Normal Form Algorithms should skip the first one and can safely continue with the simpler Negation Normal Form Algorithm on Page 46.

A Negation Normal Form Algorithm based on convolution

Considering the convolution connective requires to extend the definition of Negation Normal Form.

Definition 3.5.7
A formula of $\langle \mathcal{A}_\to, \mathcal{L}_\to, \models_t \rangle$ is said to be in *Negation Normal Form* if all negation and convolution connectives are in front of atomic subformulas such that $\neg$ and $\sim$ occur only in subformulas of the following forms A, $\neg A$, $\sim A$, or $\sim \neg A$, where A is an atom.

We present the convolution based Negation Normal Form Algorithm in two Figures. Figure 3.8 gives an overview of all rules and Figure 3.7 shows the new algorithm. It is important to realise that even if we use a semantically equivalent transformation for expressions with our implication connective, we require some semantically non-equivalent transformations. In the following, we illustrate this fact by several examples. We discuss each line of Figure 3.8 and pay particularly attention to those rules that infringe upon the semantical equivalence of the Negation Normal Form Transformation.

The first line of Figure 3.8 is concerned with the application of negation and convolution to the Truth and the False Formula. Convolution of the Truth Formula yields an expression that is mapped by all interpretations for arbitrary states to $\{\}$ and convolution of the False Formula results in an expression that is mapped by all interpretations for arbitrary states to $\{t, f\}$. We call the first expression the *None* Formula and the second expression the *Both* Formula. Syntactically, we write for the None Formula $\underset{\sim}{\perp}$ and for the Both Formula $\overline{\top}$.

The second line of Figure 3.8 shows the set of additional rules required for the application of convolution to conjuncted or disjuncted formulas. These rules require the introduction of the *consensus* and the *gullibility* operation known from [Fit94] as the meet and join operation on the approximation lattice (Page 20). Similar to the convolution connective, we restrict the use of consensus and gullibility to the Negation Normal Form Algorithm. We denote consensus and gullibility by the connectives $\otimes$ and $\oplus$, respectively and we define them with respect to two formulas F and G as follows:

$$F \otimes G \quad \overset{def}{=} \quad \sim (\sim F \vee \sim G),$$
$$F \oplus G \quad \overset{def}{=} \quad \sim (\sim F \wedge \sim G).$$

The third line of Figure 3.8, gives the rules for negated formulas containing conjunction, disjunction, gullibility, and consensus connectives.

The fourth line of Figure 3.8, shows the transformation rule for implication and rules for multiple occurrences of negation and convolution connectives.

In the fifth line of Figure 3.8, we encounter a rule that allows a transformation where the resulting formula is not semantically equivalent to the initial formula anymore. The use of this rule is linked to the last three rules of the previous line which together describe how sequences of negation and convolution connectives are reduced. This reduction is a prerequisite for the application of the rules of line six in Figure 3.8 which we discuss subsequently. First, it is crucial to understand how the reduction of sequences of negation and convolution connectives works and why we can allow a transformation of a formula F_4 that does not yield a semantically equivalent formula F_4'. The crucial point is that F_4 and F_4' are equivalent with respect to Truth Entailment.

Lemma 3.5.8
Let F be a formula of $\langle \mathcal{A}_\rightarrow, \mathcal{L}_\rightarrow, \models_t \rangle$. The following equivalence with respect to Truth Entailment holds:
$$\neg \sim F \equiv_t \neg F.$$

Input: A four-valued first-order logic formula F_4.

Proceeding: Referring to rules of Figure 3.8.

1. If the regarded formula is not in Negation Normal Form go to 2 else stop.

2. Apply rules of Part I until no rule of Part I can be applied anymore.
   If no rule of Part I can be applied go to 3.

3. Select a maximal subformula to which the rule of Part II can be applied.
   If no such subformula exists, go to 4. Otherwise apply the Part II rule to
   the selected subformula and go to 1.

4. If possible, select a maximal subformula to which a rule of Part III can
   be applied and apply this rule. Go to 1.

Output: Negation Normal Form of F_4.

Figure 3.7: Negation Normal Form Algorithm for $\langle \mathcal{A}_\rightarrow, \mathcal{L}_\rightarrow, \models_t \rangle$ based on con-
volution.

Proof of Lemma 3.5.8

$$
\begin{aligned}
    & I, \sigma \models_t \neg \sim F \\
\text{iff} \quad & I, \sigma \models_f \sim F \\
\text{iff} \quad & I, \sigma \models_f F \\
\text{iff} \quad & I, \sigma \models_t \neg F. \qquad\qquad\qquad \square
\end{aligned}
$$

Yet it is important to notice that the equivalence with respect to Truth En-
tailment is *not* a semantical equivalence as we explained in Example 3.5.6.
Therefore, we find for the rule of Part II in Figure 3.8:

$$
I, \sigma \models_f \neg \sim F \text{ iff } I, \sigma \not\models_t F
$$

but

$$
I, \sigma \models_f \neg F \text{ iff } I, \sigma \models_t F.
$$

Hence, the application of the rule of Part II causes the Negation Normal Form
F_4' of a formula F_4 to be *not* semantically equivalent to F_4. The reader familiar
with the Skolem Normal Form Transformation in classical first-order logic might
recall the question whether satisfiability or validity of a formula is preserved.
However, our case differs from that question because both satisfiability and
validity are preserved by our Negation Normal Form Algorithm. What is not
preserved is the knowledge about falsifiability of a formula. In four-valued first-
order logic falsifiability does not affect satisfiability and, thus, the Negation
Normal Form F_4' suffices to characterise F_4 for calculating Truth Entailment.
Since our purpose is the calculation of the Truth Entailment Relation, we allow
a *restricted* use of a rule that preserves satisfiability even if falsifiability is not

Part I   Semantically Equivalent Transformations

$$\dfrac{\neg\top}{\bot} \qquad \dfrac{\neg\bot}{\top} \qquad \dfrac{\neg\overline{\top}}{\overline{\top}} \qquad \dfrac{\neg\underline{\bot}}{\underline{\bot}} \qquad \dfrac{\sim\top}{\underline{\bot}} \qquad \dfrac{\sim\bot}{\overline{\top}} \qquad \dfrac{\sim\underline{\bot}}{\top} \qquad \dfrac{\sim\overline{\top}}{\bot}$$

$$\dfrac{\sim(F\wedge G)}{(\sim F\oplus\sim G)} \qquad \dfrac{\sim(F\vee G)}{(\sim F\otimes\sim G)} \qquad \dfrac{\sim(F\otimes G)}{(\sim F\vee\sim G)} \qquad \dfrac{\sim(F\oplus G)}{(\sim F\wedge\sim G)}$$

$$\dfrac{\neg(F\wedge G)}{(\neg F\vee\neg G)} \qquad \dfrac{\neg(F\vee G)}{(\neg F\wedge\neg G)} \qquad \dfrac{\neg(F\otimes G)}{(\neg F\otimes\neg G)} \qquad \dfrac{\neg(F\oplus G)}{(\neg F\oplus\neg G)}$$

$$\dfrac{(F\rightarrow G)}{(\sim F\vee G)} \qquad \dfrac{\neg\neg F}{F} \qquad \dfrac{\sim\sim F}{F} \qquad \dfrac{\neg\sim\neg\sim F}{\sim\neg\sim\neg F}$$

Part II   Semantically *Non*-Equivalent Transformation

– Application iff no rule of Part I applicable –

$$\dfrac{\neg\sim F}{\neg F}$$

Part III   Semantically Equivalent and *Non*-Equivalent Transformations

– Application iff no rule of Part I/II applicable –

$$\dfrac{\neg(\exists x)F}{(\forall x)\neg F} \qquad \dfrac{\neg(\forall x)F}{(\exists x)\neg F} \qquad \dfrac{\sim(\exists x)F}{(\forall x)\sim F} \qquad \dfrac{\sim(\forall x)F}{(\exists x)\sim F}$$

Figure 3.8: Rules of the Negation Normal Form Algorithm with convolution.

preserved. The restriction is captured in the Negation Normal Form Algorithm in Figure 3.7 by the rigorous proceeding for the application of rules.

We elucidate the importance to adhere to this strict proceeding by discussing two examples.

Example 3.5.9
With this example, we show the relationship between the rules of line five and line six in Figure 3.8. We consider the formula $\neg \sim (\forall x)F$, where F is atomic. The Negation Normal Form Algorithm does not allow a transformation through $\neg(\exists x) \sim F$ to $(\forall x)\neg \sim F$ because the rule of Part II has to be applied before. Its application leads to the formula $\neg(\forall x)F$. A further application of the rule of Part II is not possible and a rule of Part I does not become applicable. Thus, we apply rules of Part III that yield the Negation Normal Form $(\exists x)\neg F$. We verify easily that we did the right transformation:

$$
\begin{aligned}
I, \sigma &\models_t & \neg \sim (\forall x)F \ \text{ iff} \\
I, \sigma &\models_f & \sim (\forall x)F \ \text{ iff} \\
I, \sigma &\models_f & (\forall x)F \ \text{ iff} \\
I, \sigma\{x \mapsto d\} &\models_f & F \text{ for some } d \in \mathcal{D} \ \text{ iff} \\
I, \sigma\{x \mapsto d\} &\models_t & \neg F \text{ for some } d \in \mathcal{D} \ \text{ iff} \\
I, \sigma &\models_t & (\exists x)\neg F.
\end{aligned}
$$

Example 3.5.10
A similar relationship as shown above exists between rules of Part I and rules of Part II as well. We consider the formula $\neg \sim \neg \sim (\forall x)F$, where F is atomic. The Negation Normal Form Algorithm does not allow the application of the rule of Part II because its application transforms the formula to $\neg\neg \sim (\forall x)F$ and $\neg\neg(\forall x)F$. The application of a rule of Part I afterwards yields $(\forall x)F$ which is not a correct result. Hence, the Negation Normal Form Algorithm demands the application of the last rule of Part I to $\neg \sim \neg \sim (\forall x)F$ which transforms the formula to $\sim \neg \sim \neg(\forall x)F$. On this formula the rule of Part II has to be applied which yields. $\sim \neg\neg(\forall x)F$. We find that the rule for eliminating double negation connectives of Part I becomes applicable. Its application yields the formula $\sim (\forall x)F$. On this formula, neither rules of Part I nor rules of Part II can be applied and, finally, the fourth rule of Part III yields the Negation Normal Form $(\exists x) \sim F$. We verify the Negation Normal Form as follows:

$$
\begin{aligned}
I, \sigma &\models_t & \neg \sim \neg \sim (\forall x)F \ \text{ iff} & \quad (3.1) \\
I, \sigma &\models_f & \sim \neg \sim (\forall x)F \ \text{ iff} & \quad (3.2) \\
I, \sigma &\models_f & \neg \sim (\forall x)F \ \text{ iff} & \quad (3.3) \\
I, \sigma &\models_t & \sim (\forall x)F \ \text{ iff} & \quad (3.4) \\
I, \sigma &\not\models_t & (\forall x)F \ \text{ iff} & \quad (3.5) \\
I, \sigma\{x \mapsto d\} &\not\models_t & F \text{ for some } d \in \mathcal{D} \ \text{ iff} & \quad (3.6) \\
I, \sigma\{x \mapsto d\} &\models_t & \sim F \text{ for some } d \in \mathcal{D} \ \text{ iff} & \quad (3.7) \\
I, \sigma &\models_t & (\exists x) \sim F. & \quad (3.8)
\end{aligned}
$$

Obviously, the rule of Part II requires to adhere to the strict transformation proceeding of the Negation Normal Form Algorithm. But why do we use the rule of Part II at all? The requirement for the rule of Part II is linked to the rules of Part III. As we mentioned above, the reduction of sequences of negation and convolution connectives is a prerequisite for the application of quantifier rules. In the following we illustrate this fact.

Example 3.5.11
We consider the formula $\neg \sim \neg \sim (\forall x)F$ from the last example. A direct application of the quantifier rules from Part III induces the following sequence of transformation steps.

$$
\begin{array}{rl}
\neg \sim \neg \sim (\forall x) & F \\
\neg \sim \neg (\exists x) \sim & F \\
\neg \sim (\forall x) \neg \sim & F \\
\neg (\exists x) \sim \neg \sim & F \\
(\forall x) \neg \sim \neg \sim & F
\end{array}
$$

We analyse the result of this transformation as follows:

$$
\begin{array}{rl}
I, \sigma \quad \models_t & (\forall x)\neg \sim \neg \sim F \text{ iff} \\
I, \sigma\{x \mapsto d\} \quad \models_t & \neg \sim \neg \sim F \text{ for all } d \in \mathcal{D} \text{ iff} \\
I, \sigma\{x \mapsto d\} \quad \models_f & \sim \neg \sim F \text{ for all } d \in \mathcal{D} \text{ iff} \\
I, \sigma\{x \mapsto d\} \quad \models_f & \neg \sim F \text{ for all } d \in \mathcal{D} \text{ iff} \\
I, \sigma\{x \mapsto d\} \quad \models_t & \sim F \text{ for all } d \in \mathcal{D} \text{ iff} \\
I, \sigma \quad \models_t & (\forall x) \sim F.
\end{array}
$$

Obviously, the last line differs from the result (3.8) above that we got by the correct transformation.

This discrepancy shown in Example 3.5.11 does not occur with respect to Truth Entailment for the cases $\neg(Qx)F$, $\sim (Qx)F$, or $\sim \neg(Qx)F$, where Q is a quantifier. We skip a proof since this is easily verified. For the reader it is more important to understand that the rules of Part I and Part II reduce all other cases (sequences of negation and convolution connectives in front of a quantified expression) to the three cases above or to the case $(Qx)F$.

However, we encounter that considering the convolution connective in our Negation Normal Form Algorithm makes the Negation Normal Form Transformation extremely complex. As we mentioned before, an alternative option for a Negation Normal Form Algorithm exists that does not consider the convolution connective. Intuitively, we expect that such a Negation Normal Form Algorithm without convolution provides a much simpler transformation. Thus, we terminate the discussion of a Negation Normal Form Algorithm based on convolution and we consider the alternative approach in the following.

Input: A four-valued first-order logic formula F_4.

Proceeding: Referring to rules of Figure 3.10.

1. If the regarded formula is not in Negation Normal Form go to 2 else stop.

2. Apply rules of Part I until no rule of Part I can be applied anymore. If no rule of Part I can be applied go to 3.

3. If possible, select a maximal subformula to which the rule of Part II can be applied and apply this rule. Go to 1.

Output: Negation Normal Form of F_4.

Figure 3.9: Simple Negation Normal Form Algorithm for $\langle \mathcal{A}_\rightarrow, \mathcal{L}_\rightarrow, \models_t \rangle$.

A simpler Negation Normal Form Algorithm

As Example 3.5.6 shows, the formula $\neg(F \rightarrow G)$ is equivalent to $\neg F \wedge \neg G$ with respect to Truth Entailment. This equivalence with respect to Truth Entailment constitutes the basic idea for a Negation Normal Form Transformation. In order to move negation connectives inwards into a formula, we require to handle negation connectives in front of more complex formulas. With the introduction of an implication connective we have to consider additionally the case $\neg(F \rightarrow G)$. Actually, it suffices to regard a single rule for this case. The Figures 3.9 and 3.10 show the resulting Negation Normal Form Algorithm and the set of required rules, respectively.

Similar to the Negation Normal Form Algorithm based on convolution, the simpler Negation Normal Form Algorithm in Figure 3.9 demands a strict proceeding for the application of rules. The crucial idea of this proceeding is the elimination of all occurrences of double negation before the rule of Part II in Figure 3.10 is applied. The reason is that a transformation of a formula $\neg\neg(F \rightarrow G)$ to a formula of the form $\neg(\neg F \wedge \neg G)$ is not correct because $\neg\neg(F \rightarrow G) \not\equiv_t \neg(\neg F \wedge \neg G)$. We verify this easily as follows:

$$
\begin{array}{rll}
& I, \sigma & \models_t \neg(\neg F \wedge \neg G) \\
\text{iff} & I, \sigma & \models_t \neg\neg F \vee \neg\neg G \\
\text{iff} & I, \sigma & \models_t \neg\neg F \ \text{ or } \ I, \sigma \models_t \neg\neg G \\
\text{iff} & I, \sigma & \models_t F \ \text{ or } \ I, \sigma \models_t G \\
\text{not equivalent to} & I, \sigma & \not\models_t F \ \text{ or } \ I, \sigma \models_t G \\
\text{iff} & I, \sigma & \models_t \sim F \vee G \\
\text{iff} & I, \sigma & \models_t F \rightarrow G \\
\text{iff} & I, \sigma & \models_t \neg\neg(F \rightarrow G).
\end{array}
$$

Concerning the question of correctness of our simpler Negation Normal Form Algorithm, we introduce the following lemmata that we require for the proof of

Part I   Semantically Equivalent Transformations

$$\frac{\neg\neg F}{F} \qquad \frac{\neg(F\wedge G)}{(\neg F\vee\neg G)} \qquad \frac{\neg(F\vee G)}{(\neg F\wedge\neg G)}$$

$$\frac{\neg\top}{\bot} \qquad \frac{\neg\bot}{\top} \qquad \frac{\neg(\exists x)F}{(\forall x)\neg F} \qquad \frac{\neg(\forall x)F}{(\exists x)\neg F}$$

Part II   Semantically *Non*-Equivalent Transformation

– Application iff no rule of Part I applicable –

$$\frac{\neg(F\rightarrow G)}{(\neg F\wedge\neg G)}$$

Figure 3.10: Rules of the simpler Negation Normal Form Algorithm.

the correctness of the Negation Normal Form Algorithm afterwards.

Lemma 3.5.12

Let G and H be formulas. Let $F[G]$ denote a formula, where G is a subformula, and let $F[G/H]$ denote the formula $F[G]$, where G is replaced by H. We assume that all negation connectives in $F[G]$ occur in G.[6] If G is equivalent to the formula H with respect to Truth Entailment, then $F[G]$ is equivalent to $F[G/H]$ with respect to Truth Entailment.

Proof of Lemma 3.5.12

We prove Lemma 3.5.12 by structural induction. We assume that G is equivalent to H with respect to Truth Entailment.

Induction Basis: F is atomic and so it does not have subformulas other than G. Consequently, $F[G]$ is identical to G and $F[G/H]$ is identical to H. Since G is equivalent to H with respect to Truth Entailment, we find that $F[G]$ is equivalent to $F[G/H]$ with respect to Truth Entailment.

Induction Step: If $F = G$, then the claim is trivial. Thus, we assume that $F \neq G$. We have to consider many different formula structures but we find that the proof is similar for all of them. Therefore, we distinguish two different cases of formula structures and we give an example proof for one formula of each case. The proof of other formulas matching the same case is alike.

The first case covers formulas with propositional connectives including conjunction, disjunction, implication. Formulas with negation are not considered because we assume all occurrences of negation connectives to be in G. We show the proof for implication. Let $F[G]$ be of the form $F_1[G] \rightarrow F_2$. Thus, $F[G/H]$ is of the form $F_1[G/H] \rightarrow F_2$. According to the induction hypothesis, $F_1[G]$ is equivalent to $F_1[G/H]$ with respect to Truth Entailment. We find:

$$
\begin{aligned}
& I, \sigma \quad \models_t F_1[G] \rightarrow F_2 \\
\text{iff} \quad & I, \sigma \quad \not\models_t F_1[G] \ \text{ or } \ I, \sigma \models_t F_2 \\
\text{iff} \quad & I, \sigma \quad \not\models_t F_1[G/H] \ \text{ or } \ I, \sigma \models_t F_2 \\
\text{iff} \quad & I, \sigma \quad \models_t F_1[G/H] \rightarrow F_2.
\end{aligned}
$$

Let $F[G]$ be of the form $F_1 \rightarrow F_2[G]$. Thus, $F[G/H]$ is of the form $F_1 \rightarrow F_2[G/H]$. According to the induction hypothesis, $F_1[G]$ is equivalent to $F_1[G/H]$ with respect to Truth Entailment. We find:

$$
\begin{aligned}
& I, \sigma \quad \models_t F_1 \rightarrow F_2[G] \\
\text{iff} \quad & I, \sigma \quad \not\models_t F_1 \ \text{ or } \ I, \sigma \models_t F_2[G] \\
\text{iff} \quad & I, \sigma \quad \not\models_t F_1 \ \text{ or } \ I, \sigma \models_t F_2[G/H] \\
\text{iff} \quad & I, \sigma \quad \models_t F_1 \rightarrow F_2[G/H].
\end{aligned}
$$

---

[6]This is a crucial factor since $\neg\neg(F \rightarrow G) \not\equiv_t \neg(\neg F \wedge \neg G)$.

Obviously, if $F[G]$ is of the form $F_1[G] \to F_2[G]$, the claim holds as well. Hence, if G is equivalent to H with respect to Truth Entailment, then $F[G]$ is equivalent to $F[G/H]$ with respect to Truth Entailment.

The second case covers first-order connectives $\exists$ and $\forall$. We show the proof for an existentially quantified formula. Let $F[G]$ be of the form $(\exists x)F_1[G]$. Thus, $F[G/H]$ is of the form $(\exists x)F_1[G/H]$. With respect to quantified formulas, we distinguish two cases depending on the occurrences of the quantified variable. The case 2.1, where the quantified variable does not occur in G or in H, and the case 2.2, where the quantified variable occurs in both G and in H. For the imaginary case that the quantified variable occurs only in one of the subformulas G or H, we find that G and H are equivalent with respect to Truth Entailment in the trivial case that G and H are equivalent to $\bot$. Otherwise, G and H are not equivalent with respect to Truth Entailment and, thus, we do not regard this case.

In the case 2.1 the quantifier does not matter and we find according to the induction hypothesis that $F_1[G]$ is equivalent to $F_1[G/H]$ with respect to Truth Entailment. Consequently, $(\exists x)F_1[G]$ is equivalent to $(\exists x)F_1[G/H]$ with respect to Truth Entailment.

We consider the case 2.2, where the variable x occurs in G and in H. The induction hypothesis implies that if $F_1[G]$ is satisfied, then $F_1[G/H]$ is satisfied by the same interpretations I and the same states σ. This includes a state $\sigma\{x \mapsto d\}$ and, thus, we find for an interpretation I:

$$
\begin{aligned}
& I, \sigma && \models_t (\exists x)F_1[G] \\
\text{iff} \quad & I, \sigma\{x \mapsto d\} && \models_t F_1[G] \ \text{ for some } d \in \mathcal{D} \\
\text{iff} \quad & I, \sigma\{x \mapsto d\} && \models_t F_1[G/H] \ \text{ for some } d \in \mathcal{D} \\
\text{iff} \quad & I, \sigma && \models_t (\exists x)F_1[G/H]
\end{aligned}
$$

Hence, if $F_1[G]$ is equivalent to $F_1[G/H]$ with respect to Truth Entailment, then $(\exists x)F_1[G]$ is equivalent to $(\exists x)F_1[G/H]$ with respect to Truth Entailment. $\quad\square$

Lemma 3.5.13
Let F be a formula to which no rule of Part I of Figure 3.10 can be applied. If the rule of Part II of Figure 3.10 is applied to a subformula G that is maximal among those subformulas of F to which the rule of Part II is applicable, then the resulting formula F' is equivalent with respect to Truth Entailment.

Proof of Lemma 3.5.13
As shown in Example 3.5.6, $\neg(F \to G)$ is equivalent to $\neg F \wedge \neg G$ with respect to Truth Entailment. Under consideration of this equivalence with respect to Truth Entailment and Lemma 3.5.12, the claim follows immediately. $\quad\square$

Proposition 3.5.14
The Negation Normal Form Algorithm shown in the Figures 3.9 and 3.10 is correct.

Proof of Proposition 3.5.14

The claim concerning the rules of Part I follows immediately since these rules are semantically equivalent transformations as we argued in previous sections with respect to the basic four-valued first-order logic and its extension by $\top$ and $\bot$. If we apply the rules of Part I exhaustively, then we obtain the Negation Normal Form or the rule of Part II becomes applicable to a maximal subformula. Lemma 3.5.13 shows that the resulting formula is equivalent with respect to Truth Entailment. If a rule of Part I becomes applicable afterwards, then this rule does a semantically equivalent transformation and consequently, semantically equivalence with respect to Truth Entailment is preserved. Since equivalence with respect to Truth Entailment is preserved in all allowed transformation steps of the Negation Normal Form Algorithm, the Negation Normal Form of a Formula is obviously equivalent to the initial formula with respect to Truth Entailment. Hence, our second Negation Normal Form Algorithm does a correct transformation.   □

Proposition 3.5.15

Any formula of our four-valued first-order logic can be transformed to Negation Normal Form.

Proof of Proposition 3.5.15

The claim follows trivially by analysing the structure of formulas that require a transformation to Negation Normal Form. Obviously, the Negation Normal Form Algorithm shown in the Figures 3.9 and 3.10 provides a rule for any formula that is not in Negation Normal Form. Each rule eliminates negation connectives or moves them inwards. This process is continued until all negation connectives occur in front of atomic formulas.   □

Extension of the Mapping λ

For the extension of the mapping λ, we consider exclusively the formulas that are transformed by the simpler Negation Normal Form Algorithm. The reader that is interested in how formulas are mapped to classical first-order logic that are transformed by the convolution based Negation Normal Form Algorithm can find the respective mapping λ in Appendix A.

The extension of the mapping λ requires that we adapt our classical first-order logic $\langle \mathcal{A}_2, \mathcal{L}_2, \models_2 \rangle$. We extend the alphabet $\mathcal{A}_2$ by the classical implication $\rightarrow$. We denote the new alphabet by $\mathcal{A}_{2\rightarrow}$, the new language by $\mathcal{L}_{2\rightarrow}$ and the new logic by $\langle \mathcal{A}_{2\rightarrow}, \mathcal{L}_{2\rightarrow}, \models_2 \rangle$. The extended function λ is given in Figure 3.11.

Lemma 3.5.16

Theorem 3.3.3 holds for $\langle \mathcal{A}_{\rightarrow}, \mathcal{L}_{\rightarrow}, \models_t \rangle$ and $\langle \mathcal{A}_{2\rightarrow}, \mathcal{L}_{2\rightarrow}, \models_2 \rangle$, i.e., we find for a set $\mathcal{F} \in \mathcal{L}_{\rightarrow}$ and a set $\mathcal{G} \in \mathcal{L}_{\rightarrow}$ that:

$$\mathcal{F} \models_t \mathcal{G} \quad \text{iff} \quad \mathcal{F}_\lambda \models_2 \mathcal{G}_\lambda.$$

$$\lambda(F_4) \stackrel{def}{=} \begin{cases} F_4 & \text{if } F_4 \in \{\top, \, \bot\}, \\ R_+(\boldsymbol{t}) & \text{if } F_4 = R(\boldsymbol{t}), \\ R_-(\boldsymbol{t}) & \text{if } F_4 = \neg R(\boldsymbol{t}), \\ (\lambda(G) \wedge \lambda(H)) & \text{if } F_4 = (G \wedge H); \\ (\lambda(G) \vee \lambda(H)) & \text{if } F_4 = (G \vee H); \\ (\lambda(G) \rightarrow \lambda(H)) & \text{if } F_4 = (G \rightarrow H); \\ (Qx)\lambda(F) & \text{for a formula } F \text{ with } Q \text{ as quantifier.} \end{cases}$$

Figure 3.11: The extended mapping $\lambda : F_4 \mapsto F_2$ for $\langle \mathcal{A}_\rightarrow, \mathcal{L}_\rightarrow, \models_t \rangle$.

Proof of Lemma 3.5.16

We extend the proof that we gave on Page 32 by the necessary additionally induction steps. We restrict the proof to formulas in Negation Normal Form. This is possible without loss of generality because every formula of $\langle \mathcal{A}_\rightarrow, \mathcal{L}_\rightarrow, \models_t \rangle$ can be transformed to Negation Normal Form.

Only-if: Assume that I_4, σ is a model of a four-valued first-order logic formula F_4. We show that the corresponding interpretation I_2 of $\langle \mathcal{A}_{2\rightarrow}, \mathcal{L}_{2\rightarrow}, \models_2 \rangle$ and σ constitute a model of $\lambda(F_4)$.

Induction Basis: We add the cases for $\top$ and $\bot$.

7. If F_4 is of the form $\top$, then $I_4, \sigma \models_t \top$ for all I_4, σ. In this case, we find that $I_2, \sigma \models_2 \top$ for all I_2, σ and, therefore, I_2, σ is a model of $\lambda(F_4)$.

8. If F_4 is of the form $\bot$, then I_4, σ is not a model of F_4.

Induction Steps: We add the case for implication.

9. If F_4 is of the form $F \rightarrow G$, then $I_4, \sigma \not\models_t F$ or $I_4, \sigma \models_t G$. The induction hypothesis implies that $I_2, \sigma \not\models_2 \lambda(F)$ or $I_2, \sigma \models_2 \lambda(G)$ and, therefore, $I_2, \sigma \models_2 \neg\lambda(F)$ or $I_2, \sigma \models_2 \lambda(G)$ which in turn is the same as $I_2, \sigma \models_2 \lambda(F) \rightarrow \lambda(G)$. Hence, I_2, σ is a model of $\lambda(F_4)$.

If: Assume that an interpretation I_2, σ is a model of a formula F_2 in the classical first-order logic $\langle \mathcal{A}_{2\rightarrow}, \mathcal{L}_{2\rightarrow}, \models_2 \rangle$. We show that the corresponding interpretation I_4 and σ constitute a model of $\lambda^{-1}(F_2)$.

Induction Basis: We add the cases for $\top$ and $\bot$.

7. If F_2 is of the form $\top$, then $I_2, \sigma \models_2 \top$ for all I_2, σ. In this case, we find that $I_4, \sigma \models_t \top$ for all I_4, σ and, therefore, I_4, σ is a model of $\lambda^{-1}(F_2)$.

8. If F_2 is of the form $\bot$, then I_2, σ is not a model of F_2.

Induction Step: We add the case for implication.

$\supset$	$\{\}$	$\{f\}$	$\{t\}$	$\{t,f\}$
$\{\}$	$\{t\}$	$\{t\}$	$\{t\}$	$\{t\}$
$\{f\}$	$\{t\}$	$\{t\}$	$\{t\}$	$\{t\}$
$\{t\}$	$\{\}$	$\{f\}$	$\{t\}$	$\{t,f\}$
$\{t,f\}$	$\{\}$	$\{f\}$	$\{t\}$	$\{t,f\}$

Figure 3.12: Truth table of the connective $\supset$ in [AA96].

9. If F_2 is of the form $F \to G$, then $I_2, \sigma \models_2 \neg F$ or $I_2, \sigma \models_2 G$. This is the same as $I_2, \sigma \not\models_2 F$ or $I_2, \sigma \models_2 G$. The induction hypothesis implies that $I_4, \sigma \not\models_t \lambda^{-1}(F)$ or $I_4, \sigma \models_t \lambda^{-1}(G)$ which in turn is the same as $I_4, \sigma \models_t \lambda^{-1}(F) \to \lambda^{-1}(G)$. Hence, I_4, σ is a model of $\lambda^{-1}(F_2)$. $\square$

Finally, we review the Replacement Theorem for our extended four-valued first-order logic $\langle \mathcal{A}_\to, \mathcal{L}_\to, \models_t \rangle$. Obviously, the Replacement Theorem holds for $\langle \mathcal{A}_\to, \mathcal{L}_\to, \models_t \rangle$ as well. The necessary induction step to extend the proof is similar to the induction step for propositional connectives on Page 28 and, thus, we omit this extension here.

Implication Connectives in Literature

We conclude the discussion of implication connectives for four-valued first-order logic with some references to the literature. Various definitions of four-valued implication connectives can be found in [AA98, Vil02, Tso02, Mus99, AB90]. A useful definition of a paraconsistency preserving implication connective, denoted by $\supset$, is introduced in [AA96, AA98]. $\supset$ provides some properties that we require in the following chapter for the definition of four-valued Description Logics and, therefore, we describe $\supset$ closer. In this section we give a brief introduction to $\supset$ and discuss the advantage of $\to$ in comparison to $\supset$ for our four-valued first-order logic. In Section 3.8 we refer another time to $\supset$ and show the advantage of $\supset$ in comparison to $\to$ for the definition of four-valued Description Logics.

For a general understanding of $\supset$, we give in Figure 3.12 the truth table. We notice that full paraconsistency is preserved since the truth value in the lower right field of the truth table is $\{t, f\}$. The formal proof is given in the course of proving Proposition 3.8.2 on Page 62.

Another property that $\supset$ and $\to$ share is that both satisfy the Deduction Theorem which can be seen by comparing the respective truth tables with Figure 3.5. Thus, $\supset$ satisfies the first two requirements that we put in an implication connective. With respect to the third requirement, we did not find a semantically equivalent Negation Normal Form Representation[7] for arbitrary formulas with occurrences of $\supset$. The reason for this is similar as for $\to$ and becomes obvious if we consider that the description of $\supset$ by other connectives

---

[7]A representation in Negation Normal Form that is equivalent with respect to Truth Entailment can be obtained.

$\leftrightarrow$	$\{\}$	$\{f\}$	$\{t\}$	$\{t,f\}$
$\{\}$	$\{t\}$	$\{t\}$	$\{\}$	$\{\}$
$\{f\}$	$\{t\}$	$\{t,f\}$	$\{\}$	$\{f\}$
$\{t\}$	$\{\}$	$\{\}$	$\{t\}$	$\{t\}$
$\{t,f\}$	$\{\}$	$\{f\}$	$\{t\}$	$\{t,f\}$

Figure 3.13: Truth table of equivalence connective based on $\rightarrow$.

is even more complicated than the description of $\rightarrow$. In the following, we describe our attempt to represent a formula $F \supset G$ without $\supset$. We regard a functional complete set of connectives for our four-valued first-order logic including conjunction, disjunction, consensus, gullibility, the Truth Formula, the False Formula and all 24 unary involution operations[8]. We assign to two of these unary involution operations the symbols ③ and ④, and we define their semantics with respect to a formula F in the following truth table:

F	$\{\}$	$\{f\}$	$\{t\}$	$\{t,f\}$
③F	$\{\}$	$\{t\}$	$\{t,f\}$	$\{f\}$
④F	$\{t,f\}$	$\{t\}$	$\{\}$	$\{f\}$

The simplest form to describe $\supset$ requires one of these involution connectives. With respect to two formulas F and G, a formula $F \supset G$ can be expressed in the following ways:

1. ③$(F \oplus \bot) \wedge G$, or

2. ④$(F \oplus \bot) \wedge G$.

Obviously, our implication connective $\rightarrow$ provides a simpler representation in a semantically equivalent form without implication connective.

3.6 Four-valued first-order logic with Equivalence

A drawback of our implication connective $\rightarrow$ is that the connective $\leftrightarrow$ defined as $F \leftrightarrow G \stackrel{def}{=} (F \rightarrow G) \wedge (G \rightarrow F)$ does not express equivalence as we are used to understand it classically because $F \leftrightarrow G$ takes designated truth values even if the formulas F and G are not semantically equivalent. Figure 3.13 shows the respective truth table. We detect this problem for any binary connective $\oslash$ that satisfies the Deduction Theorem. A look to Figure 3.14 for $\oslash\oslash$[9] illustrates this fact. We find four cases in which an interpretation assigns to $F \oslash\oslash G$ designated truth values but where F and G are interpreted differently.

At this point, we have two options. First, we can accept the fact that equivalence in our four-valued first-order logic does not embody the classical

---

[8]There exist 24 permutations of four truth values. We assigned to each permutation a unary involution connective.

[9]$F \oslash\oslash G \stackrel{def}{=} (F \oslash G) \wedge (G \oslash F)$ and $\oslash$ defined as in Figure 3.5.

$\ominus\ominus$	$\{\}$	$\{f\}$	$\{t\}$	$\{t,f\}$
$\{\}$	$\{t\}$ or $\{t,f\}$	$\{t\}$ or $\{t,f\}$	$\{\}$ or $\{f\}$	$\{\}$ or $\{f\}$
$\{f\}$	$\{t\}$ or $\{t,f\}$	$\{t\}$ or $\{t,f\}$	$\{\}$ or $\{f\}$	$\{\}$ or $\{f\}$
$\{t\}$	$\{\}$ or $\{f\}$	$\{\}$ or $\{f\}$	$\{t\}$ or $\{t,f\}$	$\{t\}$ or $\{t,f\}$
$\{t,f\}$	$\{\}$ or $\{f\}$	$\{\}$ or $\{f\}$	$\{t\}$ or $\{t,f\}$	$\{t,f\}$

Figure 3.14: Truth table of an equivalence connective based on $\ominus$.

$\Leftrightarrow$	$\{\}$	$\{f\}$	$\{t\}$	$\{t,f\}$
$\{\}$	$\{t\}$	$\{f\}$	$\{f\}$	$\{f\}$
$\{f\}$	$\{f\}$	$\{t\}$	$\{f\}$	$\{f\}$
$\{t\}$	$\{f\}$	$\{f\}$	$\{t\}$	$\{f\}$
$\{t,f\}$	$\{f\}$	$\{f\}$	$\{f\}$	$\{t,f\}$

Figure 3.15: Truth table of the equivalence connective $\Leftrightarrow$.

concept of equivalence. Our logic does diverge from classical logics and, thus, why should we stick to classical definitions in terms of equivalence? Actually, $\leftrightarrow$ is not bad as approximation of equivalence since it distinguishes designated truth values from non-designated truth values. That is to say with respect to two formulas F and G that an interpretation assigns to $F \leftrightarrow G$ a designated truth value if and only if it assigns either to both F and G a designated truth value or to both F and G a non-designated truth value.

However, before we continue the discussion, we should think about the second option that we have, namely to find an alternative way for defining an equivalence connective. The definition of an appropriate equivalence connective is not easy at all. [AA96, AA98] introduces another implication to this end and we do the same to be able to describe an equivalence connective in a comfortable way. We define the connective $\rightsquigarrow$ with respect to two formulas F and G as:

$$F \rightsquigarrow G \stackrel{def}{=} \neg \sim \neg \sim F \vee G.$$

Further, we define $\leftrightsquigarrow$ as:

$$F \leftrightsquigarrow G \stackrel{def}{=} (F \rightsquigarrow G) \wedge (G \rightsquigarrow F)$$

and the equivalence connective $\Leftrightarrow$ as:

$$F \Leftrightarrow G \stackrel{def}{=} ((\neg F \vee G) \leftrightarrow (\neg G \vee F)) \wedge (F \leftrightsquigarrow G) \wedge \sim \neg \sim (F \leftrightsquigarrow G).$$

Figure 3.15 shows the respective truth table which illustrates that $F \Leftrightarrow G$ takes designated truth values if and only if $F^{I,\sigma} = G^{I,\sigma}$. The lower right element of the truth table is $\{t,f\}$ and, therefore, full paraconsistency is preserved for our four-valued first-order logic if we consider $\Leftrightarrow$.

We see that the definition of an equivalence connective that expresses the classical idea of equivalence in a fully paraconsistent logic is possible. Even the extension of the Negation Normal Form Algorithm that uses convolution and the mapping λ works without further complications because $\rightsquigarrow$ and $\leftrightsquigarrow$ are defined by other connectives that we introduced already.

However, we have to focus on the application of our four-valued first-order logic with respect to KRR systems. Knowledge is usually represented in classical logics and the person that encodes the knowledge probably assumes that it does not matter whether she writes a formula as $F \leftrightarrow G$ or as $(F \rightarrow G) \wedge (G \rightarrow F)$ since both formulas are semantically equivalent in classical first-order logic. Our purpose is to assign to formulas written in classical first-order logic a four-valued semantics such that inconsistencies are tolerated. That means, we consider classical first-order logic formulas as if they were formulas of our four-valued first-order logic and, thus, our four-valued first-order logic should adhere to the classical scheme for expressing equivalence. Hence, we do without an equivalence connective that expresses semantical equivalence and we consider $\leftrightarrow$ as equivalence connective in our logic.

We extend the syntax of $\langle \mathcal{A}_\rightarrow, \mathcal{L}_\rightarrow, \models_t \rangle$ by including $\leftrightarrow$ into the alphabet $\mathcal{A}_\rightarrow$. Since $\leftrightarrow$ is defined by $\rightarrow$, all theorems and propositions remain valid, and there are only minimal extensions to do with respect to the algorithms. We extend the Negation Normal Form Algorithms by a single rule, namely

$$\frac{F \leftrightarrow G}{(F \rightarrow G) \wedge (G \rightarrow F)}$$

which becomes a rule of Part I in Figure 3.8 and in Figure 3.10.

The use of $\leftrightarrow$ instead of $\Leftrightarrow$ has additionally the advantage that the simpler Negation Normal Form Algorithm can be applied.

3.7 Four-valued first-order logic with Equality

The definition of $\mathcal{SHIQ}4$ in the next Chapter requires the extension of our four-valued first-order logic by an equality predicate $\approx$. The definition of equality in a four-valued logic is not intuitive at all since classically, equality of terms is thought bivalent, i.e., either a term is equal to another term or it is not. Thus, the common definition is assigning to a formula $x \approx y$ exclusively one of the truth values $\{t\}$ or $\{f\}$. The former if and only if $x^{I,\sigma} = y^{I,\sigma}$ and the latter if and only if $x^{I,\sigma} \neq y^{I,\sigma}$. However, such a definition allows to write trivial sets of formulas as for example $\{a \approx b, \neg\, a \approx b\}$ and, thus, full paraconsistency of our logic is lost. Even initially consistent sets of formulas may become inconsistent when joining them.

Example 3.7.1
We consider the set $\{x \approx a\}$ and the set $\{x \approx b\}$, where x is a variable and a and b are constants for which we assume that $a \neq b$. Obviously, the set $\{x \approx a,\ x \approx b\}$ is inconsistent and if we interpret equality in the classical sense

then the set is trivial too.

Since our purpose is to avoid such trivialisation we require for a formula $a \approx b$ to be possibly interpreted as $\{t, f\}$. We allow such an interpretation but we have to consider that the equality predicate adheres to certain axioms of equality. Usually these are reflexivity, symmetry, transitivity, r-substitutivity[10], and f-substitutivity[11]. We do not consider f-substitutivity, because we define equality exclusively between constants. Thus, we define the axioms of equality for $\approx$ as follows.

Definition 3.7.2
The axioms of equality in our four-valued first-order logic with equality are:

reflexivity (self-identity): $(\forall x)x \approx x$,

symmetry: $(\forall x, y)(x \approx y \to y \approx x)$,

transitivity: $(\forall x, y, z)\big((x \approx y \wedge y \approx z) \to x \approx z\big)$,

r-substitutivity: $(\forall x_1, ..., x_n, y_1, ..., y_n)\big((\bigwedge_{i=1}^{n} x_i \approx y_i \ \wedge \ R(x_1, ..., x_n))$
$$\to R(y_1, ..., y_n)\big) \text{ for all } R \in \mathcal{A}_\mathcal{R},$$

where $\bigwedge_{i=1}^{n} x_i \approx y_i$ denotes the conjunction of n formulas $x_i \approx y_i$ with $x_i \in \{x_1, ..., x_n\}$ and $y_i \in \{y_1, ..., y_n\}$.

For the case that $a \approx b$ is interpreted as $\{t, f\}$, we know that $a \approx b$ holds and that we can legitimately replace b in a formula by a and vice versa. Since $a \approx b$ is interpreted as $\{t, f\}$, we know that $\neg \ a \approx b$ holds as well and by replacing b by a, we find that $\neg \ a \approx a$ which contradicts the assumption of self-identity. Nevertheless $a \approx a$ remains valid since it is interpreted as $\{t, f\}$. This suffices for our purposes in this thesis. The reader interested in how to perform arithmetic calculations in such a system is referred to [Pri00].

Concerning the syntax, we extend the alphabet $\mathcal{A}_\to$ by the infix written binary relation symbol $\approx$ for which we allow only constants as arguments. We denote the new alphabet by $\mathcal{A}_\approx$ and the corresponding extended language by $\mathcal{L}_\approx$. We call our four-valued first-order logic with equality $\langle \mathcal{A}_\approx, \mathcal{L}_\approx, \models_{t\approx} \rangle$, where $\models_{t\approx}$ denotes $\models_t$ under consideration of the axioms of equality mentioned above.

As in previous sections, we require a translation of formulas of $\langle \mathcal{A}_\approx, \mathcal{L}_\approx, \models_{t\approx} \rangle$ to a classical first-order logic. With respect to the Negation Normal Form Transformation of formulas with equality predicate, we find that $\approx$ is treated as any other binary predicate in this matter. Thus, no changes are required in the Negation Normal Form Algorithm. With respect to the mapping λ, we need some extensions.

First, we define $\langle \mathcal{A}_{2\approx}, \mathcal{L}_{2\approx}, \models_{2\approx} \rangle$ as the respective classical first-order logic by extending $\langle \mathcal{A}_{2\to}, \mathcal{L}_{2\to}, \models_2 \rangle$ with $\approx$, *diff* and the translated equational axioms. The predicate $\approx$ is the equality predicate in classical first-order logic

---

[10]Defined below.

[11]The axiom that expresses f-substitutivity is $(\forall x_1, ..., x_n, y_1, ..., y_n)\big((\bigwedge_{i=1}^{n} x_i \approx y_i) \to f(x_1, ... x_n) \approx f(y_1, ..., y_n)\big)$ for all $f \in \mathcal{A}_\mathcal{F}$.

that adheres to the translated equational axioms. *diff* is a binary predicate that accepts constants as arguments and that expresses inequality of two constants.

Second, we extend the mapping λ as follows:

$$\lambda(x \approx y) \quad \overset{def}{=} \quad x \approx y,$$
$$\lambda(\neg \ x \approx y) \quad \overset{def}{=} \quad \textit{diff}(x, y).$$

Lemma 3.7.3
Theorem 3.3.3 holds for $\langle \mathcal{A}_\approx, \mathcal{L}_\approx, \models_{t\approx} \rangle$ and $\langle \mathcal{A}_{2\approx}, \mathcal{L}_{2\approx}, \models_{2\approx} \rangle$, i.e., we find for a set $\mathcal{F} \in \mathcal{L}_\approx$ and a set $\mathcal{G} \in \mathcal{L}_\approx$ that:

$$\mathcal{F} \models_{t\approx} \mathcal{G} \ \text{ iff } \ \mathcal{F}_\lambda \models_{2\approx} \mathcal{G}_\lambda.$$

The proof of Lemma 3.7.3 requires the extension of the definition of a corresponding interpretation.

Definition 3.7.4
Let I_4 be an interpretation in our four-valued first-order logic $\langle \mathcal{A}_\approx, \mathcal{L}_\approx, \models_{t\approx} \rangle$ and let I_2 be an interpretation in $\langle \mathcal{A}_{2\approx}, \mathcal{L}_{2\approx}, \models_{2\approx} \rangle$ that has the same domain as I_4.
We say that I_4 and I_2 *correspond to each other* if for predicates other than $\approx$:

1. $I_4, \sigma \models_{t\approx} R(\boldsymbol{t})$ iff $I_2, \sigma \models_{2\approx} R_+(\boldsymbol{t})$, and

2. $I_4, \sigma \models_{f\approx} R(\boldsymbol{t})$ iff $I_2, \sigma \models_{2\approx} R_-(\boldsymbol{t})$.

and for the equality predicate $\approx$:

3. $I_4, \sigma \models_{t\approx} a \approx b$ iff $I_2, \sigma \models_{2\approx} a \approx b$, and

4. $I_4, \sigma \models_{f\approx} a \approx b$ iff $I_2, \sigma \models_{2\approx} \textit{diff}(a, b)$.

Definition 3.7.4 is only an extension of Definition 3.3.4 for the equality predicate. Similarly the proof of Lemma 3.7.3 requires only the extension of previous proofs.

Proof of Lemma 3.7.3
We extend the proof of Lemma 3.5.16 by extending the induction basis with respect to the equality predicate and under considerations of Definition 3.7.4.

Only-if: Assume that I_4, σ is a model of a four-valued first-order logic formula F_4. We show that the corresponding interpretation I_2 of $\langle \mathcal{A}_{2\approx}, \mathcal{L}_{2\approx}, \models_{2\approx} \rangle$ and σ constitute a model of $\lambda(F_4)$.

Induction Basis: We add the cases for the equality predicate $\approx$.

10. If F_4 is of the form $x \approx y$, then $I_4, \sigma \models_{t\approx} x \approx y$. In this case, we find that $I_2, \sigma \models_{2\approx} x \approx y$ and, therefore, I_2, σ is a model of $\lambda(F_4)$.

11. If F_4 is of the form $\neg\, x \approx y$, then $I_4, \sigma \models_{t\approx} \neg\, x \approx y$. In this case, we find that $I_2, \sigma \models_{2\approx} \mathit{diff}(x, y)$ and, therefore, I_2, σ is a model of $\lambda(F_4)$.

If: Assume that an interpretation I_2, σ is a model of a formula F_2 in the classical first-order logic $\langle \mathcal{A}_{2\approx}, \mathcal{L}_{2\approx}, \models_{2\approx}\rangle$. We show that the corresponding interpretation I_4 and σ constitute a model of $\lambda^{-1}(F_2)$.

Induction Basis: We add the cases for $\approx$ and *diff*.

10. If F_2 is of the form $x \approx y$, then $I_2, \sigma \models_{2\approx} x \approx y$. In this case, we find that $I_4, \sigma \models_{t\approx} x \approx y$ and, therefore, I_4, σ is a model of $\lambda^{-1}(F_2)$.

11. If F_2 is of the form $\mathit{diff}(x, y)$, then $I_2, \sigma \models_{2\approx} \mathit{diff}(x, y)$. In this case, we find that $I_4, \sigma \models_{t\approx} \neg\, x \approx y$ and, therefore, I_4, σ is a model of $\lambda^{-1}(F_2)$. $\square$

3.8   Four-valued first-order logic for $\mathcal{ALC}4$ and $\mathcal{SHIQ}4$

We require the connective $\supset$ for the definition of the semantics of four-valued Description Logics in Chapter 4. The use of $\supset$ concerns only few formulas that describe special concepts in four-valued Description Logics. We do not require the use of $\supset$ for arbitrary formulas of our four-valued first-order logic and, thus, we do not include $\supset$ as common connective into our four-valued first-order logic. If we included $\supset$ as common connective, we would permit the use of $\supset$ in arbitrary formulas which in turn causes a redefinition of the Negation Normal Form Algorithm and a redefinition of the mapping λ. This is not necessary for our purpose. In the following we show how the required formulas with $\supset$ are mapped to classical first-order logic and we clarify an important property of $\supset$, namely the following equivalence with respect to Truth Entailment:

$$\neg(F \supset G) \equiv_t F \wedge \neg G \tag{3.9}$$

To understand this equivalence, we set $\supset$ in the context of the Truth Entailment Relation and the False Entailment Relation.

19. $I, \sigma \models_t F \supset G$ iff $I, \sigma \not\models_t F$ or $I, \sigma \models_t G$,
20. $I, \sigma \models_f F \supset G$ iff $I, \sigma \models_t F$ and $I, \sigma \models_f G$.

Regarding Item 20 we verify easily that (3.9) holds because:

$$I, \sigma \models_t \neg(F \supset G)$$
$$\text{iff} \quad I, \sigma \models_f F \supset G$$
$$\text{iff} \quad I, \sigma \models_t F \text{ and } I, \sigma \models_f G$$
$$\text{iff} \quad I, \sigma \models_t F \text{ and } I, \sigma \models_t \neg G$$
$$\text{iff} \quad I, \sigma \models_t F \wedge \neg G.$$

With respect to our implication connective $\rightarrow$, we find that (3.9) does not hold. The difference of $\supset$ and $\rightarrow$ becomes clear if we set $\rightarrow$ in the context of the entailment relations $\models_t$ and $\models_f$.

21. $I, \sigma \models_t F \to G$ iff $I, \sigma \not\models_t F$ or $I, \sigma \models_t G$,
22. $I, \sigma \models_f F \to G$ iff $I, \sigma \models_f F$ and $I, \sigma \models_f G$.

Regarding the Items 19 and 21, we find that $\supset$ and $\to$ are equivalent with respect to Truth Entailment and regarding the Items 20 and 22 we find that $\supset$ and $\to$ differ with respect to False Entailment. Consequently, (3.9) does not hold with respect to our implication connective $\to$. The equivalence (3.9) becomes of special importance in Chapter 4.1 and, therefore, we determine two extended logics $\langle \mathcal{A}_\supset, \mathcal{L}_\supset, \models_t \rangle$ and $\langle \mathcal{A}_{\supset\approx}, \mathcal{L}_{\supset\approx}, \models_{t\approx} \rangle$ that consider $\supset$ and that are based on $\langle \mathcal{A}_\to, \mathcal{L}_\to, \models_t \rangle$ and $\langle \mathcal{A}_\approx, \mathcal{L}_\approx, \models_{t\approx} \rangle$, respectively.

The use of $\supset$ in $\langle \mathcal{A}_\supset, \mathcal{L}_\supset, \models_t \rangle$ and $\langle \mathcal{A}_{\supset\approx}, \mathcal{L}_{\supset\approx}, \models_{t\approx} \rangle$ is restricted such that few things change in comparison to $\langle \mathcal{A}_\to, \mathcal{L}_\to, \models_t \rangle$ and $\langle \mathcal{A}_\approx, \mathcal{L}_\approx, \models_{t\approx} \rangle$. In the logic $\langle \mathcal{A}_\supset, \mathcal{L}_\supset, \models_t \rangle$ we allow to write formulas of the form:

$$(\forall y)\big(R(x, y) \supset C[y]\big), \tag{3.10}$$

where $R(x, y)$ is an atomic formula and $C[y]$ is a formula with an occurrence of the variable y, and without occurrences of $\to$ and $\leftrightarrow$. In $\langle \mathcal{A}_{\supset\approx}, \mathcal{L}_{\supset\approx}, \models_{t\approx} \rangle$, we allow additionally formulas of the form:

$$(\forall y_1, ..., y_n)\bigg(\bigwedge_{i=1}^n \big(S(x, y_i) \wedge C[y_i]\big) \supset \bigvee_{i \neq j} y_i \approx y_j \bigg), \tag{3.11}$$

where n is an integer greater or equal than 1, $S(x, y_i)$ is an atom, $C[y_i]$ is a formula with occurrences of a variable $y_i \in \{y_1, ..., y_n\}$ and without occurrences of the connectives $\to$ and $\leftrightarrow$, $\bigwedge_{i=1}^n (S(x, y_i) \wedge C[y_i])$ stands for the conjunction of n formulas $(S(x, y_i) \wedge C[y_i])$, and $\bigvee_{i \neq j} y_i \approx y_j$ stands for the disjunction of all formulas $y_i \approx y_j$ with $i \neq j$.

In the following, we elucidate the Negation Normal Form Transformation and show how the above formulas are translated to classical first-order logic. To this end, we refer to the Negation Normal Form Algorithm in Figure 3.9 on Page 46. The transformation is linked to (3.9). We find for the negation of (3.10):

$$I, \sigma \models_t \neg(\forall y)\big(R(x, y) \supset C[y]\big)$$

iff $I, \sigma \models_f (\forall y)\big(R(x, y) \supset C[y]\big)$

iff $I, \sigma\{y \mapsto d\} \models_f R(x, y) \supset C[y]$ for some $d \in \mathcal{D}$

iff $I, \sigma\{y \mapsto d\} \models_t R(x, y)$ and $I, \sigma\{y \mapsto d\} \models_f C[y]$ for some $d \in \mathcal{D}$

iff $I, \sigma\{y \mapsto d\} \models_t R(x, y)$ and $I, \sigma\{y \mapsto d\} \models_t \neg C[y]$ for some $d \in \mathcal{D}$

iff $I, \sigma\{y \mapsto d\} \models_t R(x, y) \wedge \neg C[y]$ for some $d \in \mathcal{D}$

iff $I, \sigma \models_t (\exists y)\big(R(x, y) \wedge \neg C[y]\big)$.

We find for the negation of (3.11):

$$I, \sigma \models_t \neg(\forall y_1, ..., y_n)\left(\bigwedge_{i=1}^{n} \big(S(x, y_i) \wedge C[y_i]\big) \supset \bigvee_{i \neq j} y_i \approx y_j \right)$$

$$\text{iff} \quad I, \sigma \models_f (\forall y_1, ..., y_n)\left(\bigwedge_{i=1}^{n} \big(S(x, y_i) \wedge C[y_i]\big) \supset \bigvee_{i \neq j} y_i \approx y_j \right)$$

$$\text{iff} \quad I, \sigma\{y_1 \mapsto d, ..., y_n \mapsto d_n\} \models_f \left(\bigwedge_{i=1}^{n} \big(S(x, y_i) \wedge C[y_i]\big) \right.$$

$$\left. \supset \bigvee_{i \neq j} y_i \approx y_j \right) \text{ for some } d_1, ..., d_n \in \mathcal{D}$$

$$\text{iff} \quad I, \sigma\{y_1 \mapsto d, ..., y_n \mapsto d_n\} \models_t \bigwedge_{i=1}^{n} \big(S(x, y_i) \wedge C[y_i]\big)$$

$$\text{and} \quad I, \sigma\{y_1 \mapsto d, ..., y_n \mapsto d_n\} \models_f \bigvee_{i \neq j} y_i \approx y_j \text{ for some } d_1, ..., d_n \in \mathcal{D}$$

$$\text{iff} \quad I, \sigma\{y_1 \mapsto d, ..., y_n \mapsto d_n\} \models_t \bigwedge_{i=1}^{n} \big(S(x, y_i) \wedge C[y_i]\big)$$

$$\text{and} \quad I, \sigma\{y_1 \mapsto d, ..., y_n \mapsto d_n\} \models_t \neg \bigvee_{i \neq j} y_i \approx y_j \text{ for some } d_1, ..., d_n \in \mathcal{D}$$

$$\text{iff} \quad I, \sigma\{y_1 \mapsto d, ..., y_n \mapsto d_n\} \models_t \bigwedge_{i=1}^{n} \big(S(x, y_i) \wedge C[y_i]\big)$$

$$\text{and} \quad I, \sigma\{y_1 \mapsto d, ..., y_n \mapsto d_n\} \models_t \bigwedge_{i \neq j} \neg y_i \approx y_j \text{ for some } d_1, ..., d_n \in \mathcal{D}$$

$$\text{iff} \quad I, \sigma\{y_1 \mapsto d, ..., y_n \mapsto d_n\} \models_t \left(\bigwedge_{i=1}^{n} \big(S(x, y_i) \wedge C[y_i]\big) \right.$$

$$\left. \wedge \bigwedge_{i \neq j} \neg y_i \approx y_j \right) \text{ for some } d_1, ..., d_n \in \mathcal{D}$$

$$\text{iff} \quad I, \sigma \models_t (\exists y_1, ..., y_n)\left(\bigwedge_{i=1}^{n} \big(S(x, y_i) \wedge C[y_i]\big) \wedge \bigwedge_{i \neq j} \neg y_i \approx y_j \right).$$

A Negation Normal Form of the considered formulas is one of the following formulas:

1. $(\forall y)(R(x, y) \supset C[y])$,

2. $(\exists y)(R(x, y) \wedge C[y])$,

3. $(\forall y_1, ..., y_n)\big(\bigwedge_{i=1}^{n}(S(x, y_i) \wedge C[y_i]) \supset \bigvee_{i \neq j} y_i \approx y_j \big)$, or

4. $(\exists y_1, ..., y_n)\big(\bigwedge_{i=1}^{n}(S(x, y_i) \wedge C[y_i]) \wedge \bigwedge_{i \neq j} \neg y_i \approx y_j \big)$,

where the formulas $C[y]$ and $C[y_i]$ are in Negation Normal Form.

The translation of the first two formulas to the classical first-order logic $\langle \mathcal{A}_{2\to}, \mathcal{L}_{2\to}, \models_2 \rangle$ yields:

$$\lambda\big((\forall y)(R(x,y) \supset C[y])\big) \quad \overset{def}{=} \quad (\forall y)(R(x,y) \to \lambda(C[y])),$$

$$\lambda\big((\exists y)(R(x,y) \wedge C[y])\big) \quad \overset{def}{=} \quad (\exists y)(R(x,y) \wedge \lambda(C[y])).$$

The remaining formulas are mapped to $\langle \mathcal{A}_{2\approx}, \mathcal{L}_{2\approx}, \models_{2\approx} \rangle$ as follows:

$$\lambda\left((\forall y_1,...,y_n)\big(\bigwedge_{i=1}^{n}(S(x,y_i) \wedge C[y_i]) \supset \bigvee_{i \neq j} y_i \approx y_j\big) \right) \quad \overset{def}{=}$$

$$(\forall y_1,...,y_n)\big(\bigwedge_{i=1}^{n}(S(x,y_i) \wedge \lambda(C[y_i])) \to \bigvee_{i \neq j} y_i \approx y_j\big),$$

$$\lambda\left((\exists y_1,...,y_n)\big(\bigwedge_{i=1}^{n}(S(x,y_i) \wedge C[y_i]) \wedge \bigwedge_{i \neq j} \neg\, y_i \approx y_j\big) \right) \quad \overset{def}{=}$$

$$(\exists y_1,...,y_n)\big(\bigwedge_{i=1}^{n}(S(x,y_i) \wedge \lambda(C[y_i])) \wedge \bigwedge_{i \neq j} \mathit{diff}(y_i,y_j)\big).$$

The idea of Theorem 3.3.3 holds for $\langle \mathcal{A}_{\supset}, \mathcal{L}_{\supset}, \models_t \rangle$ and $\langle \mathcal{A}_{\supset\approx}, \mathcal{L}_{2\approx}, \models_{t\approx} \rangle$ too. Since we consider two extended logics, we reformulate the theorem as follows.

Theorem 3.8.1

The entailment of a set of formulas $\mathcal{G}$ by a set of formulas $\mathcal{F}$ in $\langle \mathcal{A}_{\supset\approx}, \mathcal{L}_{\supset\approx}, \models_{t\approx} \rangle$ can be calculated by the classical entailment relation $\models_{2\approx}$ because:

$$\mathcal{F} \models_{t\approx} \mathcal{G} \quad \text{iff} \quad \mathcal{F}_\lambda \models_{2\approx} \mathcal{G}_\lambda.$$

Proof of Theorem 3.8.1

The proof of Theorem 3.8.1 is similar to the proof of Theorem 3.3.3 on Page 32 that we extended during the proofs of Lemma 3.5.16 and Lemma 3.7.3 on the Pages 50 and 57, respectively. We take this as basis for proving Theorem 3.8.1 and, thus, we require only the extension of the previous proofs by additional induction steps.

Only-if: Assume that I_4, σ is a model of a four-valued first-order logic formula F_4 of $\langle \mathcal{A}_{\supset\approx}, \mathcal{L}_{\supset\approx}, \models_{t\approx} \rangle$. We show that the corresponding interpretation I_2 and σ constitute a model of $\lambda(F_4)$ in $\langle \mathcal{A}_{2\approx}, \mathcal{L}_{2\approx}, \models_{2\approx} \rangle$.

Additional Induction Step:

12. If F_4 is of the form $F \supset G$, then $I_4, \sigma \not\models_{t\approx} F$ or $I_4, \sigma \models_{t\approx} G$. The induction hypothesis implies that $I_2, \sigma \not\models_{2\approx} \lambda(F)$ or $I_2, \sigma \models_{2\approx} \lambda(G)$ and, therefore, $I_2, \sigma \models_{2\approx} \neg\lambda(F)$ or $I_2, \sigma \models_{2\approx} \lambda(G)$ which in turn is the same as $I_2, \sigma \models_{2\approx} \lambda(F) \to \lambda(G)$. Hence, I_2, σ is a model of $\lambda(F_4)$.

If: Assume that an interpretation I_2, σ is a model of a formula F_2 in the classical first-order logic $\langle \mathcal{A}_{2\approx}, \mathcal{L}_{2\approx}, \models_{2\approx} \rangle$. We show that the corresponding interpretation I_4 and σ constitute a model of $\lambda^{-1}(F_2)$.

Additional Induction Step:

12. If F_2 is of the form $F \rightarrow G$, then $I_2, \sigma \models_{2\approx} \neg F$ or $I_2, \sigma \models_{2\approx} G$. This is the same as $I_2, \sigma \not\models_{2\approx} F$ or $I_2, \sigma \models_{2\approx} G$. The induction hypothesis implies that $I_4, \sigma \not\models_{t\approx} \lambda^{-1}(F)$ or $I_4, \sigma \models_{t\approx} \lambda^{-1}(G)$ which in turn is the same as $I_4, \sigma \models_{t\approx} \lambda^{-1}(F) \supset \lambda^{-1}(G)$. Hence, I_4, σ is a model of $\lambda^{-1}(F_2)$.
□

Finally, we reconsider the question of paraconsistency and whether the property of full paraconsistency for the logic $\langle \mathcal{A}_{\supset\approx}, \mathcal{L}_{\supset\approx}, \models_{t\approx} \rangle$ holds. As mentioned in Section 3.4, full paraconsistency cannot be obtained if we consider the Truth and the False Formula. However, it is important to notice that $\langle \mathcal{A}_{\supset\approx}, \mathcal{L}_{\supset\approx}, \models_{t\approx} \rangle$ is fully paraconsistent if we exclude formulas that contain $\bot$ or $\top$.

Proposition 3.8.2
If we exclude the Truth and the False Formula from the alphabet $\mathcal{A}_{\supset\approx}$, then the four-valued first-order logic $\langle \mathcal{A}_{\supset\approx}, \mathcal{L}_{\supset\approx}, \models_{t\approx} \rangle$ is fully paraconsistent.

Proof of Proposition 3.8.2
For the proof of full paraconsistency, we name the logic $\langle \mathcal{A}_{\supset\approx}, \mathcal{L}_{\supset\approx}, \models_{t\approx} \rangle$ without $\bot$ and $\top$ by **L4**. We extend the proof on Page 27 by two additional induction steps for the connective $\rightarrow$ and $\supset$. The connective $\leftrightarrow$ can be described by $\rightarrow$ and $\wedge$ and, thus, it suffices to regard $\rightarrow$ and $\wedge$. Together with the proof on Page 27, the following induction steps show that arbitrary formulas F of **L4** have a model.

Additional Induction Step:

7. If F is of the form $G \rightarrow H$ and I, σ is a model of G and of H such that $G^{I,\sigma} = \{t, f\}$ and $H^{I,\sigma} = \{t, f\}$, then $(G \rightarrow H)^{I,\sigma} = \{t, f\}$. Hence, I, σ is a model of F.

8. If F is of the form $G \supset H$ and I, σ is a model of G and of H such that $G^{I,\sigma} = \{t, f\}$ and $H^{M,\sigma} = \{t, f\}$, then $(G \supset H)^{I,\sigma} = \{t, f\}$. Hence, I, σ is a model of F.

Since arbitrary formulas of **L4** have a model, we find that arbitrary sets of formulas have a model. Hence, there are no trivial sets of formulas in **L4** and, consequently, the logic **L4** is fully paraconsistent. □

From Theorem 3.8.1 and Proposition 3.8.2 follows immediately that representing a set of formulas of the logic **L4** in the classical first-order logic $\langle \mathcal{A}_{2\approx}, \mathcal{L}_{2\approx}, \models_{2\approx} \rangle$ allows paraconsistent reasoning in $\langle \mathcal{A}_{2\approx}, \mathcal{L}_{2\approx}, \models_{2\approx} \rangle$ under the semantics of **L4**.

3.9   Chapter Summary

In this chapter, we developed a four-valued first-order logic for paraconsistent reasoning by extending the approaches of [Bel76, AA98]. We put our special focus on the following issues:

1. Extension of a four-valued first-order logic by an implication connective that satisfies the Deduction Theorem and preserves full paraconsistency.

2. Transformation of four-valued first-order logic formulas with implication connective to a Negation Normal Form without implication connective.

3. Representation of paraconsistent sets of four-valued first-order logic formulas as consistent sets in classical first-order logic.

4. Calculation of the Truth Entailment Relation $\models_t$ by the classical logical consequence relation.

5. Preparation for defining four-valued Description Logics, which includes the extension of our four-valued first-order logic by an equality predicate.

Our most important conclusions are:

- The Truth Entailment Relation of four-valued first-order logic can be calculated by classical entailment in classical first-order logic and, consequently, paraconsistent reasoning capabilities are obtained in classical first-order logic under the semantics of our four-valued first-order logic (Theorem 3.3.3).

- We showed that our four-valued first-order logic extended by implication and equality is fully paraconsistent if we do not consider $\top$ and $\bot$ (Proposition 3.8.2).

- Corresponding to the extensions of our four-valued first-order logic a classical first-order logic can be determined such that paraconsistent reasoning can be described in this classical first-order logic under the semantics of our extended four-valued first-order logic (Theorem 3.8.1).

- An implication connective that satisfies the Deduction Theorem cannot be defined by the basic connectives $\wedge$, $\vee$, $\neg$.

- Semantical equivalence cannot be expressed by the classical scheme $(F \oslash G) \wedge (G \oslash F)$ and an implication connective $\oslash$ that satisfies the Deduction Theorem.

Altogether, we laid in this chapter the foundations for the development of paraconsistent Description Logics.

Chapter 4

Four-valued Description Logic

In this chapter, we discuss the relation of our four-valued first-order logics to Description Logics. In this course, two questions are of major importance to us. First, how should a four-valued $\mathcal{ALC}$ and a four-valued $\mathcal{SHIQ}$ be defined? Second, how can we transfer the methods developed for paraconsistent reasoning in classical first-order logic to realise paraconsistent reasoning in classical Description Logics?

The chapter is structured into two sections. The first section covers our four-valued version of $\mathcal{ALC}$, namely $\mathcal{ALC}4$, and the second section treats the extension of $\mathcal{ALC}4$ to $\mathcal{SHIQ}4$.

4.1   Syntax and Semantics of $\mathcal{ALC}4$

The principle ideas of the Description Logic $\mathcal{ALC}$ introduced in Chapter 2.2 are retained for $\mathcal{ALC}4$. The syntax of $\mathcal{ALC}4$ is defined like the syntax of $\mathcal{ALC}$ with the capital letters C and D denoting concepts, A denoting an atomic concept, and R denoting a role. Lower case letters a and b are used to name individuals.

We turn to the semantics of $\mathcal{ALC}4$. Our principle approach for defining the semantics of $\mathcal{ALC}4$ is the definition of a mapping φ of an $\mathcal{ALC}4$ knowledge base to a formula of our four-valued first-order logic $\langle \mathcal{A}_\supset, \mathcal{L}_\supset, \models_t \rangle$. In Chapter 2.2 Figure 2.1, we introduced a similar mapping ϕ that maps concepts of classical $\mathcal{ALC}$ to classical first-order logic formulas. The idea of φ is similar though more complicated because we cannot allow φ to map the concept $\forall R.C$ to a four-valued first-order logic formula $(\forall y)(R(x,y) \to \varphi(C,y))$. The reason for this is linked to the Negation Normal Form Transformation of quantified concepts and we demonstrate the arising problem with the following example.

Example 4.1.1
We assume that φ did the mapping of $\forall R.C$ and $\exists R.C$ to four-valued first-order logic as follows:

$$\varphi(\forall R.C, x) \ \stackrel{def}{=} \ (\forall y)\big(R(x,y) \to \varphi(C,y)\big) \text{ with } y \text{ as a new variable}, \quad (4.1)$$

$$\varphi(\exists R.C, x) \ \stackrel{def}{=} \ (\exists y)\big(R(x,y) \wedge \varphi(C,y)\big) \text{ with } y \text{ as a new variable}. \quad (4.2)$$

We find for the concept $\neg \exists R.A$, where A is an atomic concept, that it corresponds to $\neg(\exists y)(R(x, y) \wedge A(y))$ in our four-valued first-order logic. The transformation to Negation Normal Form through $(\forall y)\neg(R(x, y) \wedge A(y))$ yields $(\forall y)(\neg R(x, y) \vee \neg A(y))$. We encounter the negation of the role R and because $(R(x, y) \to A(y)) \equiv \sim R(x, y) \vee A(y)$, we cannot introduce the required implication connective by eliminating the unwanted negation. A similar problem arises with respect to a Negation Normal Form Transformation of $\neg \forall R.C$ to $\exists R.\neg C$. We explained the latter in Chapter 3.8 with respect to four-valued first-order logic. The possibility to transform a concept $\neg \exists R.C$ to a concept of the form $\forall R.\neg C$, and a concept $\neg \forall R.C$ to a concept of the form $\exists R.\neg C$ is required later for the Negation Normal Form Transformation of concepts. In Chapter 3.8 on Page 59, we introduced in Item (3.10) the formula $(\forall y)(R(x, y) \supset C[y])$ and we illustrated that a formula $\neg(\forall y)(R(x, y) \supset C[y])$ is equivalent with respect to Truth Entailment to the formula $(\exists y)(R(x, y) \wedge \neg C[y])$. This suggests to define the semantics of the concept $\forall R.C$ corresponding to $(\forall y)(R(x, y) \supset C[y])$ as follows:

$$\varphi(\forall R.C, x) \quad \overset{def}{=} \quad (\forall y)\big(R(x, y) \supset \varphi(C, y)\big) \text{ with } y \text{ as a new variable.} \quad (4.3)$$

To allow the transformation of the concept $\neg \exists R.C$ to $\forall R.\neg C$, we define the semantics of $\exists R.C$ in terms of $\forall R.C$ as follows:

$$\varphi(\exists R.C, x) \quad \overset{def}{=} \quad \varphi(\neg \forall R.\neg C, x). \quad (4.4)$$

It is worth noticing that this semantics corresponds to the definition in [Str99]. Alternative definitions for the semantics of $\forall R.C$ are proposed in [PS89] and [ML06], and we compare them briefly in the following.

The semantics proposed in [PS89], considers a definition of $\forall R.C$ based on negation and disjunction such that the formula $\varphi(\forall R.C)$ is described in four-valued first-order logic as $(\forall y)\big(\neg R(x, y) \vee \varphi(C, y)\big)$. Our mapping λ to a classical logic yields for this formula $(\forall y)\big(R_-(x, y) \vee \lambda(\varphi(C, y))\big)$ which is a formula that does not correspond to the concept $\forall R.C$ in classical $\mathcal{ALC}$.

The crucial idea in [ML06] is that roles behave bivalent in quantified concepts such that the truth values of a role are either $\{t\}$ or $\{f\}$. We do not consider this approach because our concern is assigning the semantics of a role according to a binary predicate in a four-valued first-order logic with full four-valued semantics.

The fundamentals are laid to define the entire mapping φ that gives the semantics of an $\mathcal{ALC}4$ knowledge base $(\mathcal{TB}, \mathcal{AB})$, where $\mathcal{TB}$ is a TBox and $\mathcal{AB}$ is an ABox. Figure 4.1 shows the mapping[1] φ.

A terminological axiom $C \sqsubseteq D$ is described by the connective $\to$. It is worth noticing that we obtain the same semantics by using $\supset$. The reason for this is that terminological axioms do not occur negated and the definition of $\to$ corresponds to the definition of $\supset$ with respect to Truth Entailment.

---

[1]Actually, the mapping comprises two functions $\varphi(z, x)$ and $\varphi(z)$.

$$\varphi\big((\mathcal{TB}, \mathcal{AB})\big) \stackrel{def}{=} \varphi(\mathcal{TB}) \wedge \varphi(\mathcal{AB})$$

$$\varphi(\mathcal{TB}) \stackrel{def}{=} \bigwedge_{\tau \in \mathcal{TB}} \varphi(\tau)$$

$$\varphi(\mathcal{AB}) \stackrel{def}{=} \bigwedge_{\alpha \in \mathcal{AB}} \varphi(\alpha)$$

$$\varphi(C(a)) \stackrel{def}{=} \varphi(C, a)$$

$$\varphi(R(a,b)) \stackrel{def}{=} R(a,b)$$

$$\varphi(C \sqsubseteq D) \stackrel{def}{=} (\forall x)\big(\varphi(C,x) \rightarrow \varphi(D,x)\big)$$

$$\varphi(C \equiv D) \stackrel{def}{=} (\forall x)\big(\varphi(C,x) \leftrightarrow \varphi(D,x)\big)$$

$$\varphi(\top, x) \stackrel{def}{=} \top$$

$$\varphi(\bot, x) \stackrel{def}{=} \bot$$

$$\varphi(A, x) \stackrel{def}{=} A(x)$$

$$\varphi(\neg C, x) \stackrel{def}{=} \neg\varphi(C,x)$$

$$\varphi((C \sqcap D), x) \stackrel{def}{=} (\varphi(C,x) \wedge \varphi(D,x))$$

$$\varphi((C \sqcup D), x) \stackrel{def}{=} (\varphi(C,x) \vee \varphi(D,x))$$

$$\varphi(\forall R.C, x) \stackrel{def}{=} (\forall y)\big(R(x,y) \supset \varphi(C,y)\big) \text{ with } y \text{ as a new variable}$$

$$\varphi(\exists R.C, x) \stackrel{def}{=} \varphi(\neg \forall R.\neg C, x)$$

Figure 4.1: Mapping of an $\mathcal{ALC}4$ knowledge base to four-valued first-order logic.

We use the connective $\leftrightarrow$ to describe equivalence in $\mathcal{ALC}4$. As we pointed out in Chapter 3.6, $\leftrightarrow$ does not describe semantical equivalence since such a definition of equivalence in $\mathcal{ALC}4$ is not practical for our aspired use of four-valued logics. Thus, we have to accept that semantical equivalence of the concepts C and D is not expressed by the terminological axiom $C \equiv D$ in $\mathcal{ALC}4$. However, equivalence with respect to Truth Entailment is expressed and calculating inferences in a classical logic corresponds only to Truth Entailment anyway.

It is worth noticing that our definition of subsumption and equivalence makes $\mathcal{ALC}4$ different to the four-valued $\mathcal{ALC}$s introduced in [PS89, Str97]. Both [PS89] and [Str97] define subsumption of two concepts C and D in the same way, with respect to a four-valued interpretation that assigns a *positive extension* and a *negative extension* to C and to D. The idea of the four-valued interpretation in Description Logics corresponds to the idea of our four-valued interpretation I introduced in Chapter 3.2. The so-called positive extension corresponds to the set R_+^I that I assigns to a relation symbol R and the so-called negative extension corresponds to the set R_-^I. Adopting our notation, the four-valued interpretation in [PS89, Str97] assigns to a concept C a positive extension C_+^I comprising those individuals of the interpretation's domain that belong to the concept C, and a negative extension C_-^I contains those individuals of the domain that are known not to belong to C. Both extensions may overlap and there might be individuals in the domain that are neither in C_+^I nor in C_-^I. Obviously, it sounds strange if an individual a belongs to the concept C because it is in C_+^I, and at the same time a does not belong to C because it is in C_-^I as well. However, this is nothing else than assigning to the instance $C(a)$ the truth value $\{t, f\}$. Now, [PS89, Str97] define the subsumption of the concept C by the concept D such that the positive extension of C is a subset of the positive extension of D and the negative extension of D is a subset of the negative extension of C. We write for this subsumption in the following $C \sqsubseteq_s D$.

With respect to our semantics, we find that D *subsumes* C (written $C \sqsubseteq D$) if and only if the positive extension of C is a subset of the positive extension of D disregarding the negative extensions. This is due to the fact that we define subsumption based on the implication connective $\rightarrow$ which describes the Truth Entailment Relation $\models_t$ but not the False Entailment Relation $\models_f$. Consequently, there exist interpretations that satisfy $C \sqsubseteq D$ but that do not satisfy $C \sqsubseteq_s D$. We exemplify this in the following.

Example 4.1.2
We assume two concepts *GermanWriter* and *German*, and an interpretation I that determines a single individual *franzKafka* to belong to the concepts *GermanWriter* and *German*, and at the same time I determines *franzKafka* not to belong to the concept *German*. Furthermore, we do not know an individual that does not belong to the concept *GermanWriter* such that its negative extension is the empty set. Thus, we find $GermanWriter_+^I \subseteq German_+^I$ but $German_-^I \not\subseteq GermanWriter_-^I$. Consequently, I satisfies the subsumption $GermanWriter \sqsubseteq German$ but I does not satisfy $GermanWriter \sqsubseteq_s German$.

4.2 Reasoning with $\mathcal{ALC}$4

In Chapter 2.2, we introduced the various inference tasks in Description Logics and we showed that satisfiability, equivalence, and disjointness can be expressed exclusively by subsumption in $\mathcal{ALC}$. Accordingly, our subsumption allows to perform these inference tasks in $\mathcal{ALC}$4. We define:

1. a concept C is *unsatisfiable* if and only if $C \sqsubseteq \bot$;

2. two concepts C and D are *equivalent* if and only if $C \sqsubseteq D$ and $D \sqsubseteq C$;

3. two concepts C and D are *disjoint* if and only if $(C \sqcap D) \sqsubseteq \bot$.

From the semantic point of view we understand satisfiability as follows. We refer to φ and we say that a concept C is *satisfiable* if and only if there exist an interpretation I and a state σ in our four-valued first-order logic that satisfy $\varphi(C, x)$. In this case we call I, σ a *model* of C. This is the same as observing with respect to the semantics of [PS89, Str97] that the positive extension of C is not empty. Regarding an ABox $\mathcal{AB}$ and a TBox $\mathcal{TB}$, we say that $\mathcal{AB}$ and $\mathcal{TB}$ are satisfied by I and σ if and only if I, σ is a model of $\varphi(\mathcal{AB})$ and $\varphi(\mathcal{TB})$, respectively. If I, σ is a model of both $\varphi(\mathcal{AB})$ and $\varphi(\mathcal{TB})$, then I, σ is a *model* of the $\mathcal{ALC}$4 knowledge base $(\mathcal{TB}, \mathcal{AB})$. We say that a knowledge base $(\mathcal{TB}, \mathcal{AB})$ *entails* a concept C if all models of $(\mathcal{TB}, \mathcal{AB})$ are models of C. In this case, we write $(\mathcal{TB}, \mathcal{AB}) \models_t C$.

The instance checking problem that we defined for $\mathcal{ALC}$ translates directly to $\mathcal{ALC}$4. An assertion $C(a)$ (respectively $R(a, b)$) is *entailed* by an ABox $\mathcal{AB}$ if every interpretation that satisfies $\mathcal{AB}$ also satisfies $C(a)$ (respectively $R(a, b)$). We write $\mathcal{AB} \models_t C(a)$ (respectively $\mathcal{AB} \models_t R(a, b)$).

The calculation of subsumption is performed in classical $\mathcal{ALC}$. Thus, we translate an $\mathcal{ALC}$4 knowledge base to a representation in classical $\mathcal{ALC}$. For this translation, we have to elaborate three things. First, we require the classical $\mathcal{ALC}$ to which we translate expressions of $\mathcal{ALC}$4. We define this classical $\mathcal{ALC}$ to be like the $\mathcal{ALC}$ introduced in Chapter 2.2 and we determine additionally that it provides for each concept C in $\mathcal{ALC}$4 two concepts C_+ and C_-, and for each role R in $\mathcal{ALC}$4 a role R_+. Second, we have to transform concepts of $\mathcal{ALC}$4 to Negation Normal Form. We say that a concept is in Negation Normal Form if all negations in it occur in front of atomic concepts only. The Negation Normal Form Algorithm is given in Figure 4.2. Third, we define the mapping λ for $\mathcal{ALC}$4 that transforms assertions, terminological axioms, and concepts to classical $\mathcal{ALC}$. Figure 4.3 shows λ.

It is worth noticing that this mapping λ does the same transformation as the mapping λ for four-valued first-order logic. The relation is depicted in Figure 4.4. This means that if we translate an $\mathcal{ALC}$4 knowledge base $\mathcal{KB}_4$ to the $\mathcal{ALC}$ knowledge base $\mathcal{KB}_2$, we obtain the same result as for the transformation of $\mathcal{KB}_4$ to four-valued first-order logic and further to classical first-order logic and, finally, mapping the representation in classical first-order logic to $\mathcal{ALC}$ again. We illustrate this fact in the following example.

Input: An $\mathcal{ALC}4$ concept C_4.
Proceeding: Apply the rules below until Negation Normal Form is established. Every rule has two parts, the upper part and the lower part. A rule is applied by replacing an instance of the upper part by an instance of the lower part.
Output: Negation Normal Form of C_4.

$$\frac{\neg\neg C}{C} \qquad \frac{\neg(C\sqcap D)}{(\neg C\sqcup\neg D)} \qquad \frac{\neg(C\sqcup D)}{(\neg C\sqcap\neg D)} \qquad \frac{\neg\exists R.C}{\forall R.\neg C} \qquad \frac{\neg\forall R.C}{\exists R.\neg C}$$

Figure 4.2: Negation Normal Form Algorithm for concepts of $\mathcal{ALC}4$.

Mapping for assertions α

$$\lambda(\alpha) \stackrel{def}{=} \begin{cases} \lambda(C)(a) & \text{if } \alpha = C(a), \\ R_+(a,b) & \text{if } \alpha = R(a,b). \end{cases}$$

Mapping for terminological axioms τ

$$\lambda(\tau) \stackrel{def}{=} \begin{cases} \lambda(C) \sqsubseteq \lambda(D) & \text{if } \tau = C \sqsubseteq D, \\ \lambda(C) \equiv \lambda(D) & \text{if } \tau = C \equiv D. \end{cases}$$

Mapping for concepts C_4

$$\lambda(C_4) \stackrel{def}{=} \begin{cases} C_4 & \text{if } C_4 \in \{\top, \bot\}, \\ A_+ & \text{if } C_4 = A, \\ A_- & \text{if } C_4 = \neg A, \\ (\lambda(C) \sqcup \lambda(D)) & \text{if } C_4 = (C \sqcup D), \\ (\lambda(C) \sqcap \lambda(D)) & \text{if } C_4 = (C \sqcap D), \\ \exists R_+.\lambda(C) & \text{if } C_4 = \exists R.C, \\ \forall R_+.\lambda(C) & \text{if } C_4 = \forall R.C. \end{cases}$$

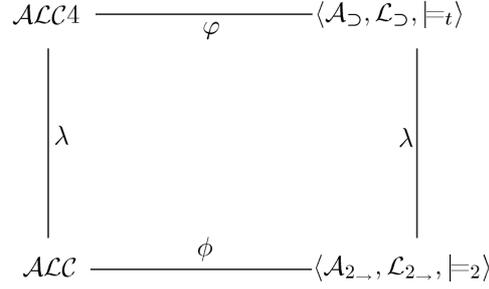Figure 4.3: Mapping λ for $\mathcal{ALC}4$ to $\mathcal{ALC}$.

$$\mathcal{ALC}4 \xrightarrow{\quad\varphi\quad} \langle \mathcal{A}_\supset, \mathcal{L}_\supset, \models_t \rangle$$

$$\lambda \Big\downarrow \qquad\qquad\qquad\qquad \Big\downarrow \lambda$$

$$\mathcal{ALC} \xrightarrow{\quad\phi\quad} \langle \mathcal{A}_{2\rightarrow}, \mathcal{L}_{2\rightarrow}, \models_2 \rangle$$

Figure 4.4: Relation of $\mathcal{ALC}4$ to $\mathcal{ALC}$, four-valued first-order logic and classical first-order logic.

Example 4.2.1

We consider the concept $\neg\exists R.A$. This concept is translated to four-valued first-order logic as $\neg\neg(\forall y)\big(R(x,y) \supset \neg A(y)\big)$ which has the Negation Normal Form $(\forall y)\big(R(x,y) \supset \neg A(y)\big)$. A mapping to classical first-order logic yields $(\forall y)\big(R_+(x,y) \rightarrow A_-(y)\big)$ which corresponds to the $\mathcal{ALC}$ concept $\forall R_+.A_-$.

For the direct mapping from $\mathcal{ALC}4$ to $\mathcal{ALC}$, we consider the Negation Normal Form of the concept $\neg\exists R.A$ which is $\forall R.\neg A$. Obviously, the mapping to $\mathcal{ALC}$ yields the concept $\forall R_+.A_-$.

We can establish this relation trivially for any expression in an $\mathcal{ALC}4$ knowledge base. Thus, we skip a proof.

Paraconsistent Reasoning in $\mathcal{ALC}$

With the previously developed formalisms, paraconsistent reasoning in classical $\mathcal{ALC}$ becomes possible. The crucial idea concerning the application of $\mathcal{ALC}4$ is that we consider knowledge bases encoded in classical $\mathcal{ALC}$ as if they were written in $\mathcal{ALC}4$. This means, we interpret axioms and assertions of a classical knowledge base by our four-valued semantics. The transformation of all concepts to Negation Normal Form and the mapping λ of terminological axioms and assertions provides the instrument to transform a paraconsistent knowledge base to a consistent representation in classical $\mathcal{ALC}$. With respect to this consistent representation, classical reasoning algorithms become applicable. We exemplify this in the following.

Example 4.2.2

We define two knowledge bases $(\mathcal{TB}_1, \mathcal{AB}_1)$ and $(\mathcal{TB}_2, \mathcal{AB}_2)$:

$\mathcal{TB}_1$ comprises the axioms
$$\begin{aligned} CzechWriter &\equiv (Czech \sqcap Writer), \\ CzechWriter &\equiv \neg GermanWriter; \end{aligned}$$

$\mathcal{AB}_1$ comprises the assertion
$$Czech(franzKafka);$$

$\mathcal{TB}_2$ comprises the axiom
$$GermanWriter \quad \equiv \quad (German \sqcap Writer);$$

$\mathcal{AB}_2$ comprises the assertions
$$German(franzKafka),$$
$$Writer(franzKafka).$$

In classical $\mathcal{ALC}$ the union $(\mathcal{TB}_1 \cup \mathcal{TB}_2, \mathcal{AB}_1 \cup \mathcal{AB}_2)$ of both knowledge bases is unsatisfiable because Franz Kafka is a German writer and a Czech writer but these two concepts exclude each other. We interpret $(\mathcal{TB}_1, \mathcal{AB}_1)$ and $(\mathcal{TB}_2, \mathcal{AB}_2)$ to be written in $\mathcal{ALC}4$ and, thus, $(\mathcal{TB}_1 \cup \mathcal{TB}_2, \mathcal{AB}_1 \cup \mathcal{AB}_2)$ is satisfiable. However, we want to use classical reasoning systems to perform reasoning tasks on $(\mathcal{TB}_1 \cup \mathcal{TB}_2, \mathcal{AB}_1 \cup \mathcal{AB}_2)$ and, therefore, a consistent representation in classical $\mathcal{ALC}$ is required. We transform both knowledge bases to the corresponding representation in classical $\mathcal{ALC}$, namely $(\mathcal{TB}_{1\lambda}, \mathcal{AB}_{1\lambda})$ for the first knowledge base and $(\mathcal{TB}_{2\lambda}, \mathcal{AB}_{2\lambda})$ for the second knowledge base. $\mathcal{TB}_{1\lambda}$, $\mathcal{AB}_{1\lambda}$, $\mathcal{TB}_{2\lambda}$, and $\mathcal{AB}_{2\lambda}$ have the following form:

$\mathcal{TB}_{1\lambda}$ comprising the axioms
$$CzechWriter_+ \quad \equiv \quad (Czech_+ \sqcap Writer_+),$$
$$CzechWriter_+ \quad \equiv \quad GermanWriter_-;$$

$\mathcal{AB}_{1\lambda}$ comprising the assertion
$$Czech_+(franzKafka);$$

$\mathcal{TB}_{2\lambda}$ comprising the axiom
$$GermanWriter_+ \quad \equiv \quad (German_+ \sqcap Writer_+);$$

$\mathcal{AB}_{2\lambda}$ comprising the assertions
$$German_+(franzKafka),$$
$$Writer_+(franzKafka).$$

Both knowledge bases can be joined by a classical KRR system to a single knowledge base $(\mathcal{TB}_{1\lambda} \cup \mathcal{TB}_{2\lambda}, \mathcal{AB}_{1\lambda} \cup \mathcal{AB}_{2\lambda})$ that is satisfiable in $\mathcal{ALC}$. As example for the logical consequences of $(\mathcal{TB}_{1\lambda} \cup \mathcal{TB}_{2\lambda}, \mathcal{AB}_{1\lambda} \cup \mathcal{AB}_{2\lambda})$, we obviously find that

$$(\mathcal{TB}_{1\lambda} \cup \mathcal{TB}_{2\lambda}, \mathcal{AB}_{1\lambda} \cup \mathcal{AB}_{2\lambda}) \models_2 Writer_+(franzKafka)$$

and[2]

$$(\mathcal{TB}_{1\lambda} \cup \mathcal{TB}_{2\lambda}, \mathcal{AB}_{1\lambda} \cup \mathcal{AB}_{2\lambda}) \not\models_2 Irish_+(franzKafka).$$

---

[2]Just to give an example that not every formula is entailed by the knowledge base.

4.3 Syntax and Semantics of $\mathcal{SHIQ}4$

The principle ideas of the Description Logic $\mathcal{SHIQ}$ are retained for $\mathcal{SHIQ}4$. Thus, we adopt the syntax of $\mathcal{SHIQ}$ that we introduced in Chapter 2.2 for $\mathcal{SHIQ}4$, where $\mathcal{A_R}$ denotes the set of role names, $\mathcal{RB}$ denotes a RBox, $\mathcal{TB}$ denotes a TBox, and $\mathcal{AB}$ denotes an ABox.

The semantics to $\mathcal{SHIQ}4$ is based on the semantics of $\mathcal{ALC}4$ defined by the mapping φ of Figure 4.1. From now on, φ maps to the four-valued first-order logic $\langle \mathcal{A}_{\supset\approx}, \mathcal{L}_{\supset\approx}, \models_{t\approx} \rangle$ and is extended as follows according to the additional constructors that $\mathcal{SHIQ}$ provides:

$$\varphi\big((\mathcal{RB}, \mathcal{TB}, \mathcal{AB})\big) \;\stackrel{def}{=}\; \varphi(\mathcal{RB}) \wedge \varphi(\mathcal{TB}) \wedge \varphi(\mathcal{AB})$$

$$\varphi(\mathcal{RB}) \;\stackrel{def}{=}\; \bigwedge_{\rho \in \mathcal{RB}} \varphi(\rho) \wedge \bigwedge_{R \in \mathcal{A_R}} \varphi(R)$$

$$\varphi(a \approx b) \;\stackrel{def}{=}\; a \approx b,$$

$$\varphi(a \not\approx b) \;\stackrel{def}{=}\; \neg\, a \approx b,$$

$$\varphi(R) \;\stackrel{def}{=}\; (\forall x, y)\big(R(x, y) \leftrightarrow R^-(y, x)\big),$$

$$\varphi(R_1 \sqsubseteq R_2) \;\stackrel{def}{=}\; (\forall x, y)\big(R_1(x, y) \to R_2(x, y)\big),$$

$$\varphi(Trans(R)) \;\stackrel{def}{=}\; (\forall x, y, z)\big(R(x, y) \wedge R(y, z) \to R(x, z)\big),$$

$$\varphi(\leq n\,S.C, x) \;\stackrel{def}{=}\; (\forall y_1, ..., y_{n+1})\Big(\bigwedge_{i=1}^{n+1} \big(S(x, y_i) \wedge \varphi(C, y_i)\big) \supset \bigvee_{i \neq j} y_i \approx y_j \Big),$$

$$\varphi(\geq n\,S.C, x) \;\stackrel{def}{=}\; \begin{cases} \varphi(\neg \forall S.\neg C, x) & \text{if } n = 1, \\ \varphi(\neg \leq (n-1)S.C, x) & \text{otherwise.} \end{cases}$$

where y_i, y_j are new variables. The semantics of the above extension is very similar to the two-valued case. We encounter a difference with respect to qualified number restriction because the corresponding four-valued first-order logic formulas employ the $\supset$ connective. This semantics of qualified number restriction differs to the semantics in [PS89] due to the same reasons as mentioned with respect to $\forall R.C$ in the previous section, where we defined the semantics of $\exists R.C$ by $\neg \forall R.\neg C$. The same reasons force us to define semantics of $\geq n\,S.C$ by $\neg \leq (n-1)S.C$.

In Chapter 2.2, we introduced two equivalences with respect to qualified number restriction. Since we defined the semantics of $\geq n\,S.C$ in terms of $\leq n\,S.C$, these equivalences hold in $\mathcal{SHIQ}4$ as well. We find:

$$\neg \geq n\,S.C \;\equiv\; \begin{cases} \forall S.\neg C & \text{if } n = 1, \\ \leq (n-1)S.C & \text{otherwise;} \end{cases} \tag{4.5}$$

$$\neg \leq n\,S.C \;\equiv\; \geq (n+1)S.C. \tag{4.6}$$

4.4 Reasoning with $\mathcal{SHIQ}4$

For the representation of $\mathcal{SHIQ}4$ in a classical Description Logic, we refer to a classical $\mathcal{SHIQ}$ similar to the one that we introduced in Chapter 2.2. We determine that the classical $\mathcal{SHIQ}$ provides for each concept C in $\mathcal{SHIQ}4$ two concepts C_+ and C_-, for each role R in $\mathcal{SHIQ}4$ a role R_+, for the equality predicate $\approx$ the identical predicate in $\mathcal{SHIQ}$, and for the inequality predicate the predicate $diff$.

We extend the Negation Normal Form Algorithm of $\mathcal{ALC}4$ to $\mathcal{SHIQ}4$ by adding the following transformation rules:

$$\frac{\neg \geq 1 S.C}{\forall S. \neg C} \qquad \frac{\neg \geq n S.C}{\leq (n-1) S.C} \qquad \frac{\neg \leq n S.C}{\geq (n+1) S.C}$$

Finally, we define the mapping λ for assertions, terminological axioms, transitivity axioms, role inclusion axioms, and concepts of $\mathcal{SHIQ}4$ to $\mathcal{SHIQ}$. Figure 4.5 shows the entire mapping λ. This mapping λ is a simple extension of the mapping λ for $\mathcal{ALC}4$. An interesting question remains concerning the newly introduced concepts of qualified number restriction. The mapping λ maps qualified number restriction in $\mathcal{SHIQ}4$ directly to qualified number restriction in $\mathcal{SHIQ}$. If we dare a closer look, we encounter a serious problem in doing so. The reason is that the concept $\geq n\,S.C$ disguises an inequality predicate that is not transformed to the predicate $diff$. We illustrate this fact as follows.

Example 4.4.1
We consider the concept $\geq n\,S.C$ where C shall be atomic. This concept corresponds to the formula

$$\neg(\forall y_1, ..., y_n)\bigg(\bigwedge_{i=1}^{n}\big(S(x, y_i) \wedge C(y_i)\big) \supset \bigvee_{i \neq j} y_i \approx y_j\bigg)$$

in four-valued first-order logic. As we illustrated in Chapter 3.8, the Negation Normal Form of this formula is

$$(\exists y_1, ..., y_n)\bigg(\bigwedge_{i=1}^{n}\big(S(x, y_i) \wedge C(y_i)\big) \wedge \bigwedge_{i \neq j} \neg\, y_i \approx y_j\bigg).$$

The mapping to classical first-order logic yields

$$(\exists y_1, ..., y_n)\bigg(\bigwedge_{i=1}^{n}\big(S(x, y_i) \wedge C(y_i)\big) \wedge \bigwedge_{i \neq j} diff(y_i, y_j)\bigg).$$

A mapping of this formula to $\mathcal{SHIQ}$ is not possible because the predicate $diff(y_i, y_j)$ is not defined within the concept of qualified number restriction. The $\mathcal{SHIQ}$ concept $\geq n\,S.C$ corresponds to the classical first-order logic formula

$$(\exists y_1, ..., y_n)\bigg(\bigwedge_{i=1}^{n}\big(S(x, y_i) \wedge C(y_i)\big) \wedge \bigwedge_{i \neq j} \neg\, y_i \approx y_j\bigg).$$

Mapping for assertions α

$$\lambda(\alpha) \;\stackrel{def}{=}\; \begin{cases} a \approx b & \text{if } \alpha = a \approx b, \\ \mathit{diff}(a,b) & \text{if } \alpha = a \not\approx b, \\ \lambda(C)(a) & \text{if } \alpha = C(a), \\ R_+(a,b) & \text{if } \alpha = R(a,b). \end{cases}$$

Mapping for RBox axioms ρ

$$\lambda(\rho) \;\stackrel{def}{=}\; \begin{cases} R_{1+} \sqsubseteq R_{2+} & \text{if } \tau = R_1 \sqsubseteq R_2, \\ Trans(R_+) & \text{if } \tau = Trans(R). \end{cases}$$

Mapping for terminological axioms τ

$$\lambda(\tau) \;\stackrel{def}{=}\; \begin{cases} \lambda(C) \sqsubseteq \lambda(D) & \text{if } \tau = C \sqsubseteq D, \\ \lambda(C) \equiv \lambda(D) & \text{if } \tau = C \equiv D. \end{cases}$$

Mapping for concepts C_4

$$\lambda(C_4) \;\stackrel{def}{=}\; \begin{cases} C_4 & \text{if } C_4 \in \{\top,\ \bot\}, \\ A_+ & \text{if } C_4 = A, \\ A_- & \text{if } C_4 = \neg A, \\ (\lambda(C) \sqcup \lambda(D)) & \text{if } C_4 = (C \sqcup D), \\ (\lambda(C) \sqcap \lambda(D)) & \text{if } C_4 = (C \sqcap D), \\ \exists R_+.\lambda(C) & \text{if } C_4 = \exists R.C, \\ \forall R_+.\lambda(C) & \text{if } C_4 = \forall R.C, \\ \geq n\, S_+.\lambda(C) & \text{if } C_4 =\; \geq n\, S.C, \\ \leq n\, S_+.\lambda(C) & \text{if } C_4 =\; \leq n\, S.C. \end{cases}$$

Figure 4.5: Mapping λ for $\mathcal{SHIQ}4$ to $\mathcal{SHIQ}$.

The issue illustrated in Example 4.4.1 has several consequences. First, we observe that the relation in Figure 4.4 does not hold for $\mathcal{SHIQ}4$. Second, we encounter that we might introduce inconsistencies in $\mathcal{SHIQ}$ by translating concepts with qualified number restriction from $\mathcal{SHIQ}4$ to $\mathcal{SHIQ}$. A simple example is the translation of $(\geq 3\, S.A \sqcap\ \leq 2\, S.A)$ to $(\geq 3\, S_+.A_+ \sqcap\ \leq 2\, S_+.A_+)$. In $\mathcal{SHIQ}$, there cannot be three or more individuals that are in the concept A_+ and, simultaneously, at most two individuals that are in the concept A_+.

For our goal to perform paraconsistent reasoning in $\mathcal{SHIQ}$, we find only one solution to this problem. We require an alternative definition of the semantics of qualified number restriction. This semantics should enable the representation of $\geq n\, S.A$ and $\leq n\, S.A$ in $\mathcal{SHIQ}$ such that the role S and the concept A in $\geq n\, S.A$ are mapped differently to $\mathcal{SHIQ}$ than the role S and the concept A in $\leq n\, S.A$. However, this proposal should be a motivation for further development since the definition of an alternative semantics goes beyond the scope of this thesis.

We conclude this chapter by remarking that we obtained a paraconsistent four-valued version of $\mathcal{SHIQ}$. Since we defined the semantics with respect to a mapping to four-valued first-order logic, we find that $\mathcal{SHIQ}4$ is fully paraconsistent if we disregard the concepts $\top$ and $\bot$. However, our mapping to classical $\mathcal{SHIQ}$ does not translate all paraconsistent knowledge bases of $\mathcal{SHIQ}4$ to consistent knowledge bases in $\mathcal{SHIQ}$. Thus, we obtained only a limited ability for paraconsistent reasoning in classical $\mathcal{SHIQ}$ under the semantics of $\mathcal{SHIQ}4$.

4.5   Chapter Summary

In this chapter, we transferred results of Chapter 3 to a setting of four-valued Description Logic. We turned our major attention to the following issues:

1. How should a four-valued $\mathcal{ALC}$ and a four-valued $\mathcal{SHIQ}$ be defined?

2. How can we transfer the methods developed for paraconsistent reasoning in classical first-order logic to realise paraconsistent reasoning in classical Description Logics?

We adapted the Negation Normal Form Algorithm and the mapping λ to Description Logics. We discussed the relation of our four-valued first-order logics to four-valued Description Logics and we defined a four-valued $\mathcal{ALC}$ and a four-valued $\mathcal{SHIQ}$. We compared our approach to the approaches of [PS89] and [Str97].

The most important conclusions are:

- Under consideration of a mapping from a four-valued Description Logic to a classical Description Logic, the semantics of the concepts $\forall R.C$ and $\leq n\, S.C$ should be defined differently than in [PS89]. A practical definition for $\forall R.C$ is provided in [Str97]. We showed that $\leq n\, S.C$ can be defined

similarly. The implication connective $\supset$ of [AA96] is advantageous for a semantic mapping to four-valued first-order logic.

- Our definition of the semantics of four-valued Description Logics by the semantic mapping φ allows to transfer valuable properties of four-valued first-order logics to four-valued Description Logics. Thus, our four-valued Description Logics are fully paraconsistent if we disregard the concepts $\perp$ and $\top$.

- We found that our four-valued $\mathcal{ALC}$ can be translated to classical $\mathcal{ALC}$ such that paraconsistent reasoning becomes possible in classical $\mathcal{ALC}$.

- With respect to our four-valued $\mathcal{SHIQ}$, we found that the mapping of a knowledge base to classical $\mathcal{SHIQ}$ might trivialise this knowledge base because the translation of qualified number restriction is problematic. We see a solution for this problem by considering an alternative semantics for qualified number restriction. As starting point for investigations we propose [ML06], where a solution for un-qualified number restriction is obtained.

Chapter 5

The Realisation in KAON2

This chapter illustrates an application of the previously developed theory. We consider the ontology management infrastructure KAON2 of the University of Karlsruhe[1] and we describe how paraconsistent reasoning capabilities may be implemented for KAON2. The actual implementation is not object of this thesis. However, we conclude with a small example based on a first experimental implementation. This example illustrates the advantage of our approach and captures our initial motivation of merging two ontologies.

5.1 From Description Logics to OWL

A comprehensive introduction to the Web Ontology Language (*OWL*) is provided by the World Wide Web Consortium (W3C)[2] in [SWM04]. For reasons of self-containedness we recapitulate in the following the crucial ideas and notation of OWL used in this thesis.

Description Logics constitute a basis of the Web Ontology Language. The principle idea of OWL is to provide a formalism for defining and instantiating Web ontologies. The term ontology is borrowed from philosophy, where the term refers to the science of describing entities in the world and how they are related. An *OWL ontology* may include descriptions of *classes*, *properties* and their *instances* that we can identify with concepts, roles, and individuals in Description Logic, respectively. Thus, an OWL ontology is similar to a knowledge base in Description Logic.

The syntax that we use to write OWL ontologies is the OWL/XML syntax defined in [HEPS03].

Example 5.1.1
The concept assertion *GermanWriter(franzKafka)* is written as follows:

```
<owlx:Individual owlx:name="franzKafka">
    <owlx:type owlx:name="#germanWriter" />
</owlx:Individual>
```

---

[1] http://kaon2.semanticweb.org
[2] http://www.w3.org.

The terminological axiom $GermanWriter \sqsubseteq (German \sqcap Writer)$ translates to OWL/XML as follows:

```
<owlx:SubClassOf>
  <owlx:sub>
    <owlx:Class owlx:name="#germanWriter" />
  </owlx:sub>
  <owlx:super>
    <owlx:IntersectionOf>
      <owlx:Class owlx:name="#german" />
      <owlx:Class owlx:name="#writer" />
    </owlx:IntersectionOf>
  </owlx:super>
</owlx:SubClassOf>
```

The terminological axiom $CzechWriter \equiv \neg GermanWriter$ has the form:

```
<owlx:EquivalentClasses>
  <owlx:Class owlx:name="#czechWriter" />
  <owlx:ComplementOf>
    <owlx:Class owlx:name="#germanWriter" />
  </owlx:ComplementOf>
</owlx:EquivalentClasses>
```

5.2 Proposal for an Implementation

KAON2 is an ontology management infrastructure implemented in Java 1.5 and targeted for business applications. It includes tools for ontology management and provides a framework for building ontology-based applications. An important focus of KAON2 is scalable and efficient reasoning with ontologies.

Some tools of the KAON2 infrastructure are exposed in the KAON2 OWL Tools package. We propose to extend the OWL Tools by adding a class with the required algorithms to transform axioms of an OWL ontology to a so-called *paraconsistent representation*. This turns an inconsistent ontology[3] into a consistent ontology. In the following, we describe where an implementation might be embedded in the OWL Tools package and how this is linked to the object oriented structure of KAON2.

Object Oriented Structure

The OWL Tools provide the necessary instrument for parsing an OWL ontology and its axioms. In KAON2, every axiom is represented by a Java class and implements the methods that it inherits from an interface called `Axiom`. In this context, the method `accept(KAON2Visitor visitor)` is crucial because parsing ontology axioms is realised through the *Visitor*[4] design pattern

---

[3]We assume this ontology to be paraconsistent with respect to a four-valued semantics.

[4]We assume that the reader interested in the concrete implementation disposes of basic knowledge in object oriented design.
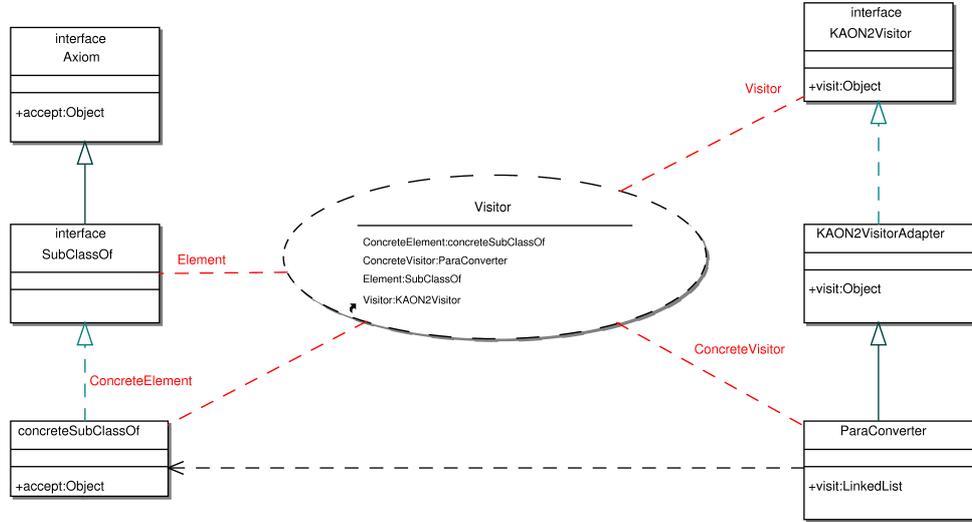
Figure 5.1: KAON2Visitor design pattern extended by Class ParaConverter.

[Kno02, GHJV95]. Figure 5.1 shows an example for a SubClassOf axiom. Every axiom represents a *Concrete Element* in the context of the design pattern. An axiom accepts any object that is an instance of a *Concrete KAON2Visitor*, e.g., an instance of the Class `KAON2VisitorAdapter`. For our purposes, the implementation of such a Concrete KAON2Visitor provides us with the means to perform the required transformation on every axiom. Thus, we propose to extend the Class `KAON2VisitorAdapter` by the Class `ParaConverter` that comprises the entire code for the transformation of an axiom to a paraconsistent representation.

Application Example

As mentioned above, the implementation is not in the scope of this thesis. However, we want to provide a motivating example that is based on a first experimental implementation.

The crucial idea is to refer to existing ontologies that should be merged. We assume that these ontologies are encoded in classical logic style and that merging them yields an inconsistent ontology. Thus, we assign a four-valued semantics under which an inconsistent ontology becomes paraconsistent. We translate the ontologies to a consistent representation in classical logic that allows to merge the ontologies by preserving consistency. Consequently, KAON2 is enabled to perform reasoning tasks on the resulting ontology.

We consider the ontology $\mathcal{E}_1$ and the ontology $\mathcal{E}_2$. The Figures 5.2 and 5.3 show the respective ontologies.

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>

<owlx:Ontology owlx:name="example0_input1"
    xml:base="example0_input1"
    xmlns:owlx="http://www.w3.org/2003/05/owl-xml#">

<owlx:Individual owlx:name="franzKafka">
  <owlx:type owlx:name="#czech" />
</owlx:Individual>

<owlx:Individual owlx:name="franzKafka">
    <owlx:type owlx:name="#writer" />
</owlx:Individual>

<owlx:EquivalentClasses>
  <owlx:Class owlx:name="#czechWriter" />
  <owlx:ComplementOf>
    <owlx:Class owlx:name="#germanWriter" />
  </owlx:ComplementOf>
</owlx:EquivalentClasses>

<owlx:EquivalentClasses>
  <owlx:Class owlx:name="#czechWriter" />
  <owlx:IntersectionOf>
    <owlx:Class owlx:name="#czech" />
    <owlx:Class owlx:name="#writer" />
  </owlx:IntersectionOf>
</owlx:EquivalentClasses>

</owlx:Ontology>
```

Figure 5.2: Ontology $\mathcal{E}_1$.

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<owlx:Ontology owlx:name="example0_input2"
    xml:base="example0_input2"
    xmlns:owlx="http://www.w3.org/2003/05/owl-xml#">

<owlx:Individual owlx:name="franzKafka">
    <owlx:type owlx:name="#germanWriter" />
</owlx:Individual>

<owlx:SubClassOf>
  <owlx:sub>
    <owlx:Class owlx:name="#germanWriter" />
  </owlx:sub>
  <owlx:super>
    <owlx:IntersectionOf>
      <owlx:Class owlx:name="#german" />
      <owlx:Class owlx:name="#writer" />
    </owlx:IntersectionOf>
  </owlx:super>
</owlx:SubClassOf>

<owlx:SubClassOf>
  <owlx:sub>
    <owlx:IntersectionOf>
      <owlx:Class owlx:name="#german" />
      <owlx:Class owlx:name="#writer" />
    </owlx:IntersectionOf>
  </owlx:sub>
  <owlx:super>
    <owlx:Class owlx:name="#germanWriter" />
  </owlx:super>
</owlx:SubClassOf>

</owlx:Ontology>
```

Figure 5.3: Ontology $\mathcal{E}_2$.

In the following, we give a representation of both ontologies in Description Logic:

$$
\begin{aligned}
\mathcal{E}_1 \;=\; \{ &Czech(franzKafka),\; Writer(franzKafka), \\
&CzechWriter \equiv \neg GermanWriter, \\
&CzechWriter \equiv (Czech \sqcap Writer) \},
\end{aligned}
$$

$$
\begin{aligned}
\mathcal{E}_2 \;=\; \{ &GermanWriter(franzKafka), \\
&GermanWriter \sqsubseteq (German \sqcap Writer), \\
&(German \sqcap Writer) \sqsubseteq GermanWriter \}.
\end{aligned}
$$

Both ontologies are consistent if we regard them separately. However, merging both ontologies to a single ontology yields an inconsistent ontology such that KAON2 cannot work with it.

Based on our proposed extension of the OWL Tools, KAON2 is enabled to read and transform both ontologies to consistent representations that remain consistent when they are merged. The result of the combined ontologies is given in Figure 5.4 as ontology $\mathcal{E}3$.

The representation in Description Logic of the ontology $\mathcal{E}3$ is the following:

$$
\begin{aligned}
\mathcal{E}_3 \;=\; \{ &Czech_+(franzKafka),\; Writer_+(franzKafka), \\
&CzechWriter_+ \equiv GermanWriter_-, \\
&GermanWriter_+(franzKafka), \\
&CzechWriter_+ \equiv (Czech_+ \sqcap Writer_+) \} \\
&GermanWriter_+ \sqsubseteq (German_+ \sqcap Writer_+), \\
&(German_+ \sqcap Writer_+) \sqsubseteq GermanWriter_+ \}.
\end{aligned}
$$

Since $\mathcal{E}_3$ is consistent, KAON2 can perform reasoning tasks on the combined ontologies $\mathcal{E}_1$ and $\mathcal{E}_2$. Queries of a user that refer to the syntax of $\mathcal{E}_1$ and $\mathcal{E}_2$ have to be transformed corresponding to the transformation that is done with the axioms of the initial ontologies. The result of the query should be re-transformed to the initial syntax.

5.3   Chapter Summary

In this chapter, we illustrated how paraconsistent reasoning can be realised in the ontology management infrastructure KAON2. Our major concern was:

1. To exemplify that our theory is applicable with respect to existing KRR systems.

2. To motivate a further implementation of paraconsistent reasoning capabilities for KAON2 and other KRR systems.

We proposed a proceeding how to extend KAON2 with paraconsistent reasoning capabilities. We discussed an example based on two ontologies that are

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE owlx:Ontology [ <!ENTITY a 'example0_para#'> ]>

<owlx:Ontology owlx:name="http://kaon2.semanticweb.org/
                                         example0_para"
    xml:base="http://kaon2.semanticweb.org/example0_para"
    xmlns:owlx="http://www.w3.org/2003/05/owl-xml#">

<owlx:Individual owlx:name="franzKafka">
    <owlx:type owlx:name="&a;czech+"/>
    <owlx:type owlx:name="&a;germanWriter+"/>
    <owlx:type owlx:name="&a;writer+"/>
</owlx:Individual>

<owlx:EquivalentClasses>
    <owlx:Class owlx:name="&a;czechWriter+"/>
    <owlx:Class owlx:name="&a;germanWriter-"/>
</owlx:EquivalentClasses>

<owlx:EquivalentClasses>
    <owlx:Class owlx:name="&a;czechWriter+"/>
    <owlx:IntersectionOf>
        <owlx:Class owlx:name="&a;czech+"/>
        <owlx:Class owlx:name="&a;writer+"/>
    </owlx:IntersectionOf>
</owlx:EquivalentClasses>

<owlx:SubClassOf>
    <owlx:sub>
        <owlx:IntersectionOf>
            <owlx:Class owlx:name="&a;german+"/>
            <owlx:Class owlx:name="&a;writer+"/>
        </owlx:IntersectionOf>
    </owlx:sub>
    <owlx:super>
        <owlx:Class owlx:name="&a;germanWriter+"/>
    </owlx:super>
</owlx:SubClassOf>

<owlx:Class owlx:name="&a;germanWriter+" owlx:complete="false">
    <owlx:IntersectionOf>
        <owlx:Class owlx:name="&a;german+"/>
        <owlx:Class owlx:name="&a;writer+"/>
    </owlx:IntersectionOf>
</owlx:Class>

</owlx:Ontology>
```

Figure 5.4: Ontology $\mathcal{E}_3$.

inconsistent if a classical semantics is assumed and the ontologies are merged. We showed that assuming our four-valued semantics allows us to consider inconsistent ontologies as paraconsistent ontologies. These paraconsistent ontologies can be transformed to consistent ontologies such that merging them yields a consistent ontology on which KAON2 can perform reasoning tasks. The most important conclusions are:

- Paraconsistent reasoning can be realized for KRR systems that work with classical logics, e.g., KAON2.

- Non-trivial inferences can be calculated with respect to inconsistent ontologies.

Chapter 6

Conclusions and Related Work

In this last chapter, we resume the results of this thesis and we provide an outlook for future and related developments. The section about related work sets our thesis in relation to other approaches in the field of inconsistency handling.

6.1 Conclusions

In this thesis we discussed in detail how paraconsistent reasoning can be performed in classical logics under the semantics of four-valued logics. We extended Belnap's four-valued logic by an implication connective $\rightarrow$ that satisfies the Deduction Theorem with respect to the Truth Entailment Relation $\models_t$ and that is representable in classical first-order logic by classical implication. Based on [Avr99], we discussed that any connective that satisfies the Deduction Theorem is non-monotonic with respect to $\leq_k$ and, thus, it cannot be defined by $\wedge$, $\vee$, $\neg$. Furthermore, we found that an implication connective that satisfies the Deduction Theorem cannot be used to express semantical equivalence as in classical first-order logic. The comparison of our implication connective $\rightarrow$ with the implication connective $\supset$ of [AA96, AA98] showed that a formula $F \rightarrow G$ has a simple semantically equivalent representation, while the representation of $F \supset G$ in a semantically equivalent form is more complex. In contrast, $\supset$ is preferable for the definition of the concept $\forall R.C$ in four-valued Description Logics. The definition of subsumption in four-valued Description Logics can be done by both implication connectives in the same way.

The main result of this thesis is that paraconsistent reasoning in our extended four-valued first-order logic with implication and equality is obtained in classical first-order logic (Theorem 3.8.1). Furthermore, excluding a Truth and a False Formula, we showed that our extended four-valued first-order logic is fully paraconsistent (Proposition 3.8.2).

We transferred the results for four-valued first-order logic to a setting for Description Logics. The property of (full) paraconsistency translates to our four-valued Description Logics. We defined four-valued versions of $\mathcal{ALC}$ and $\mathcal{SHIQ}$ and we compared our approach to the approaches in [PS89, Str97]. An important result is that our definition of subsumption differs from [PS89,

Str97] since our purpose is the mapping of terminological axioms with a four-valued semantics to a simple representation in a classical logic. Our definition of subsumption allows a simple description by classical subsumption.

Furthermore, we encountered that mapping a knowledge base of our four-valued $\mathcal{SHIQ}$ to classical $\mathcal{SHIQ}$ might trivialise this knowledge base because the translation of qualified number restriction is problematic.

The motivation of this thesis is the desire to obtain paraconsistent reasoning capabilities for existing KRR systems that work with classical logics. The work presented here contributes to this goal by extending the fundamentals and investigating the opportunities for a concrete implementation. Our proposal is the extension of the OWL Tools by implementing our Negation Normal Form Algorithm and the mapping λ. We exemplified that this allows paraconsistent reasoning with KAON2.

6.2   Future Work

An open issue is the implementation of our algorithms for KAON2 and other KRR systems. With respect to such an implementation, we suggest a further investigation of paraconsistent Description Logics. Our approach covers only basic ideas but for the application with respect to OWL, four-valued Description Logics with data types as for example four-valued versions of $\mathcal{SHIQ}(D)$ or $\mathcal{SHOIN}(D)$ should be elaborated. A first approach to a four-valued version of $\mathcal{SHOIN}(D)$ is given in [ML06] and we describe it below and in Section 6.3.

Concerning the development of four-valued Description Logics, we found that a thorough analysis of particular properties of Description Logics is essential. For instance, a four-valued semantics for the concept $\forall R.C$ can be defined in different ways. The first definition was given in [PS89] other definitions are given in [Str97] and in [ML06]. All three definitions have a justification with respect to a special application of four-valued Description Logics. We showed that the definition of [Str97] is advantageous for our mapping to classical Description Logics. In [Str97], a comparison to the semantics of [PS89] is made. A detailed comparison with the semantics of [ML06] does not exists yet. The crucial idea in [ML06] is that roles behave bivalent in quantified concepts such that the truth values of a role are either $\{t\}$ or $\{f\}$. We did not consider this approach since our concern is assigning the semantics of a role according to a binary predicate in a four-valued first-order logic with full four-valued semantics. However, considering the semantics of concepts as four-valued but the semantics of roles as two-valued suffices to establish paraconsistency in $\mathcal{ALC}$[1] because contradictions are not expressible.

Example 6.2.1
Assume the concept $(\exists R.C)$ and its negation $\neg(\exists R.C)$. We transform the latter to $(\forall R.\neg C)$. Obviously, if there is an interpretation that assigns to the role R

---

[1]However, for paraconsistent number restriction, a four-valued semantics of roles is required.

the truth value $\{t\}$ and to the concept C the truth value $\{t, f\}$, we find a model of $\{(\exists R.C), \neg(\exists R.C)\}$. Thus, it is not necessary for a role to take the truth value $\{t, f\}$ for obtaining a paraconsistent Description Logic.

This induces that achieving a fully paraconsistent Description Logic requires less effort than to obtain a fully paraconsistent first-order logic. Another fact makes this obvious. We developed a four-valued implication connective such that the set $\{F \rightarrow G, \neg(F \rightarrow G)\}$ has a model. With respect to Description Logics, our implication describes subsumption. However, negated terminological axioms do not occur in Description Logics. Hence, the subsumption relation for a paraconsistent Description Logic might be designed differently.

With respect to a four-valued $\mathcal{SHIQ}$, we propose the elaboration of an alternative semantics for qualified number restriction. This semantics should allow a translation of a knowledge base to classical $\mathcal{SHIQ}$ that cannot become trivial due to concepts with qualified number restriction. As starting point for investigations we propose [ML06], where a solution for un-qualified number restriction is obtained.

Altogether, we suggest as future work the investigation of different semantics for paraconsistent Description Logics and to compare the expressivity and applicability of the resulting Description Logics.

6.3   Related Work

In [BHS05b], an introduction to several approaches for dealing with inconsistencies and many references are given. We illustrate two recent approaches for inconsistency handling that are different to the paraconsistent approach and we put afterwards the focus on the paraconsistent approach. In Chapter 1, we already gave a general overview about literature concerning paraconsistency. Thus, the reader that is interested in other approaches to paraconsistency than the four-valued approach finds the references in Chapter 1. Below, we consider mainly paraconsistency with four-valued logics and we introduce some of the most important works.

Alternative Approaches to Inconsistency Handling

One of the alternative approaches is to consider consistent sub-sets selected according to some preference criteria [HvHtT05, BBC03, HV05, VVS05]. We sketch the principle idea of the first reference in a few lines. In [HvHtT05], a selection function is proposed that chooses consistent sub-theories of an inconsistent set such that classical reasoning systems can be applied to draw inferences on the consistent sub-theories. An interesting fact in this approach is that no predefined preference criterion is required because the selection function depends on individual queries. [HvHtT05] illustrates the proceeding with respect to a prototype implementation called PION.

A second approach to deal with inconsistencies is the attempt to detect them and to somehow "repair" the inconsistencies. References to this approach are

[Sch05, SC03], where a method is proposed to identify minimal sets of axioms which need to be removed to resolve an inconsistency.

Paraconsistent Approaches to Inconsistency Handling

The principle idea of the paraconsistent approaches for inconsistency handling is considering a new semantics that tolerates inconsistencies. In this thesis, we took this approach with respect to a four-valued semantics. In the following, we emphasise some of the most important related work in the field of four-valued logics.

A related approach concerning four-valued first-order logic describe the work of Arieli and Avron that we cited many times in this thesis. The most important articles for us are [AA96] and [AA98], where a detailed discussion of four-valued logics is given that extend Belnap's four-valued logic. An major achievement is the definition of an implication connective $\supset$ that induces a Gentzen-type Sequent calculus and a Hilbert-type system. Based on $\supset$, another implication connective $\rightarrow$[2] is defined and the relationship to $\supset$ and to material implication[3] is discussed. In [AA98, Ari98, AA97, AA95, AA94], Arieli and Avron put the focus on the discussion of different bilattice based logics and the refinement of its entailment relations. The idea is to consider only some preferred models to draw conclusions. Several preference criteria are discussed. Finally, a discussion about advantages of a four-valued logic to a three-valued logic is given. A related discussion about the expressive power of three-valued and four-valued logics can be found in [Avr99]. A comparison of different multi-valued logics is provided in [CMM05, CS96], where the main focus is put on the complexity of inferences.

In [AD02, AD01], Arieli and Denecker work on modelling paraconsistent reasoning in classical logic. A major concern of their work is to describe preferential reasoning based on multiple-valued logic by circumscription-like axioms in a classical logic. The method that Arieli and Denecker propose to map formulas of a four-valued logic to a classical logic inspired our work. However, [AD02, AD01] do not consider implication or equivalence connectives in the four-valued logic. An alternative approach for mapping a four-valued logic to classical logic is given in [RR98], where Belnap's four-valued logic is translated to classical first-order logic.

The first approaches to four-valued semantics for Description Logics were made in [PS89, PS88] by Patel-Schneider. His work refers to a very early form of a Description Logic for systems that perform reasoning based on structural subsumption algorithms. Thus, several constructors of today's Description Logics are not considered, namely disjunction ($\sqcup$), full negation ($\neg$), and full existential quantification ($\exists R.C$). The major concern of Patel-Schneider is the investigation of a new semantics for an expressive Description Logic with tractable subsumption. In [PS89], Patel-Schneider discusses that his four-valued semantics supports an interesting subsumption relationship that is easily computable

---

[2]Different to our implication $\rightarrow$ with $A \rightarrow B \stackrel{def}{=} \sim A \vee B$, the implication $\rightarrow$ in [AA96] is defined as $A \rightarrow B \stackrel{def}{=} (A \supset B) \wedge (\neg B \supset \neg A)$.

[3]With the term *material implication*, we refer to an implication $\mapsto$ for which holds that the formula $F \mapsto G$ is semantically equivalent to $\neg F \vee G$.

and that is useful in several domains. In [PS88], Patel-Schneider emphasises a four-valued identity relationship that supports the definition of number restriction.

Based on the four-valued semantics of [PS89], Meghini and Straccia determine in [MS96] a four-valued version of an $\mathcal{ALC}$-like Description Logic called $\mathcal{ALC}$MIR. The principle idea is to obtain a *relevance* Description Logic that provides better computational behaviour than classical Description Logics. Differing to [PS89], Meghini and Straccia propose a new semantics for the concept $\forall R.C$. With respect to this new semantics, the then structural subsumption algorithms do not work [Str97]. Thus, a Gentzen-type sequent calculus for $\mathcal{ALC}$MIR is introduced that supports the new semantics. In [Str97], Straccia discusses the differences between the four-valued semantics in [MS96] and [PS89] by defining two versions of a four-valued $\mathcal{ALC}$. The first is based on the semantics introduced in [MS96] that provides an entailment relation $\models_4^A$. The second version of $\mathcal{ALC}$ is based on the semantics of [PS89] with the entailment relation $\models_4^B$. Straccia shows that $\models_4^B \subset \models_4^A$ and, thus, that the semantics in [MS96] allows more inferences. A comprehensive discussion of all the details is provided in the PhD thesis of Straccia [Str99] about logic based multimedia document retrieval.

An approach to *constructive* paraconsistent Description Logics is made in [WOS03]. The principle idea is to obtain inconsistency tolerant Description Logics by using a constructive negation instead of classical negation and intuitionistic implication instead of classical implication. Three different constructive Description Logics and corresponding Hilbert-type and tableaux systems are introduced.

A related approach to our thesis with respect to mapping four-valued Description Logics to classical Description Logics is made by Ma and Lin in [ML06]. Ma and Lin develop a four-valued $\mathcal{SHOIN}(D)$ based on ideas of Arieli and Avron for the definition of subsumption. In [ML06], three subsumption relations are introduced. The first corresponds to material implication in a four-valued first-order logic and the other subsumption relations correspond to the implication connectives $\supset$ and $\rightarrow$ of [AA96, AA98]. It is suggested to use existing knowledge bases and to divide the terminological axioms in three groups according to the different subsumption relations. Finally, a method is proposed to represent all three subsumption relations in classical $\mathcal{SHOIN}(D)$ by classical subsumption. The approach of [ML06] has the advantage to exploit the expressivity of a four-valued logic in four-valued Description Logics. However, we perceive a problem for practical application since the division of terminological axioms into the three different groups has to be done by hand.

6.4 Acknowledgements

Yue Ma, Graham Priest, Peter Patel-Schneider and Heinrich Wansing for inspiration concerning diverse concepts of paraconsistent reasoning. I am thankful to Ozan Kahramanoğullari for informal discussions and critical questions that helped me to shape my thoughts. Many thanks to my friends, who gave me feedback to a preliminary version of this work.

This thesis is dedicated to my family, who always made sure that I feel their support.

Appendix A

Proofs

Equivalences in four-valued first-order logic

The table of valid equivalences is shown in figure A.1. We give the proof for two equivalences as example. The other equivalences can be proven likewise.

Proof of $\quad \neg(F \wedge G) \equiv \neg F \vee \neg G$.
The formula $\neg(F \wedge G)$ is semantically equivalent to the formula $\neg F \vee \neg G$ if and only if we find for all interpretations I and all states σ that $(\neg(F \wedge G))^{I,\sigma} = (\neg F \vee \neg G)^{I,\sigma}$. This in turn is the case if and only if we find for all interpretations I and all states σ that

1. $I, \sigma \models_t \neg(F \wedge G)$ iff $I, \sigma \models_t \neg F \vee \neg G$ and

2. $I, \sigma \models_f \neg(F \wedge G)$ iff $I, \sigma \models_f \neg F \vee \neg G$.

We prove both items beginning with Item 1. Assume for all interpretations I and all states σ that $I, \sigma \models_t \neg(F \wedge G)$. This is the case if and only if

$$I, \sigma \models_f F \wedge G$$
$$\text{iff} \quad I, \sigma \models_f F \text{ or } I, \sigma \models_f G$$
$$\text{iff} \quad I, \sigma \models_t \neg F \text{ or } I, \sigma \models_t \neg G$$
$$\text{iff} \quad I, \sigma \models_t \neg F \vee \neg G.$$

With respect to Item 2, assume for all interpretations I and all states σ that $I, \sigma \models_f \neg(F \wedge G)$. This is the case if and only if

$$I, \sigma \models_t F \wedge G$$
$$\text{iff} \quad I, \sigma \models_t F \text{ and } I, \sigma \models_t G$$
$$\text{iff} \quad I, \sigma \models_f \neg F \text{ and } I, \sigma \models_f \neg G$$
$$\text{iff} \quad I, \sigma \models_f \neg F \vee \neg G.$$

$\square$

Proof of $\quad \neg(\exists x)F \equiv (\forall x)\neg F$.
The formula $\neg(\exists x)F$ is semantically equivalent to the formula $(\forall x)\neg F$ if and

only if we find for all interpretations I and all states σ that $(\neg(\exists x)F)^{I,\sigma} = ((\forall x)\neg F)^{I,\sigma}$. This in turn is the case if and only if we find for all interpretations I and all states σ that

1. $I, \sigma \models_t \neg(\exists x)F$ iff $I, \sigma \models_t (\forall x)\neg F$ and

2. $I, \sigma \models_f \neg(\exists x)F$ iff $I, \sigma \models_f (\forall x)\neg F$.

We prove both items beginning with Item 1. Assume for all interpretations I and all states σ that $I, \sigma \models_t \neg(\exists x)F$. This is the case if and only if

$$I, \sigma \models_f (\exists x)F$$
$$\text{iff} \quad I, \sigma\{x \mapsto d\} \models_f F \text{ for all } d \in \mathcal{D}$$
$$\text{iff} \quad I, \sigma\{x \mapsto d\} \models_t \neg F \text{ for all } d \in \mathcal{D}$$
$$\text{iff} \quad I, \sigma \models_t (\forall x)\neg F.$$

With respect to Item 2, assume for all interpretations I and all states σ that $I, \sigma \models_f \neg(\exists x)F$. This is the case if and only if

$$I, \sigma \models_t (\exists x)F$$
$$\text{iff} \quad I, \sigma\{x \mapsto d\} \models_t F \text{ for some } d \in \mathcal{D}$$
$$\text{iff} \quad I, \sigma\{x \mapsto d\} \models_f \neg F \text{ for some } d \in \mathcal{D}$$
$$\text{iff} \quad I, \sigma \models_f (\forall x)\neg F.$$

$\square$

Deduction Theorem in four-valued first-order logic

Theorem A.1
Let $\mathcal{F}$ be a set of formulas and let F and G be formulas of $\langle \mathcal{A}_\rightarrow, \mathcal{L}_\rightarrow, \models_t \rangle$. We find that:

$$\mathcal{F} \cup \{F\} \models_t G \text{ iff } \mathcal{F} \models_t F \rightarrow G$$

Proof of Theorem A.1
Only-if: We assume that $\mathcal{F} \cup \{F\} \models_t G$ holds. In order for $\mathcal{F} \models_t F \rightarrow G$ to hold, all models of $\mathcal{F}$ have to be models of $F \rightarrow G$ as well. Let I, σ be a model of $\mathcal{F}$. If I, σ is a model of F, then it is a model of $\mathcal{F} \cup \{F\}$ and by the assumption it is a model of G too. By the definition of $\rightarrow$, we find that if I, σ is a model of F and of G, then it is a model of $F \rightarrow G$. If I, σ is not a model of F, then I, σ is a model of $F \rightarrow G$ by the definition of $\rightarrow$.
If: We assume that $\mathcal{F} \models_t F \rightarrow G$ holds. In order for $\mathcal{F} \cup \{F\} \models_t G$ to hold, all models of $\mathcal{F} \cup \{F\}$ have to be models of G as well. Let I, σ be a model of $\mathcal{F} \cup \{F\}$, then it is a model of $F \rightarrow G$ by the assumption. In order for I, σ to be a model of $\mathcal{F} \cup \{F\}$, I, σ has to be a model of F. With respect to the definition of $\rightarrow$, we find that if I, σ is a model of F and of $F \rightarrow G$, then I, σ is a model of G as well. $\square$

$$\neg\neg F \equiv F$$

$$F \wedge F \equiv F$$
$$F \vee F \equiv F$$

$$F \wedge G \equiv G \wedge F$$
$$F \vee G \equiv G \vee F$$

$$(F \wedge G) \wedge H \equiv F \wedge (G \wedge H)$$
$$(F \vee G) \vee H \equiv F \vee (G \vee H)$$

$$(F \wedge G) \vee F \equiv F$$
$$(F \vee G) \wedge F \equiv F$$

$$F \wedge (G \vee H) \equiv (F \wedge G) \vee (F \wedge H)$$
$$F \vee (G \wedge H) \equiv (F \vee G) \wedge (F \vee H)$$

$$\neg(F \wedge G) \equiv \neg F \vee \neg G$$
$$\neg(F \vee G) \equiv \neg F \wedge \neg G$$

$$\neg(\exists x)F \equiv (\forall x)\neg F$$
$$\neg(\forall x)F \equiv (\exists x)\neg F$$

$$(\forall x)F \wedge (\forall x)G \equiv (\forall x)(F \wedge G)$$
$$(\exists x)F \vee (\exists x)G \equiv (\exists x)(F \vee G)$$

$$(\exists x)(\exists y)F \equiv (\exists y)(\exists x)F$$
$$(\forall x)(\forall y)F \equiv (\forall y)(\forall x)F$$

Figure A.1: Valid equivalences in four-valued first-order logic.

$$\lambda(F_4) \stackrel{def}{=} \begin{cases} \top & \text{if } F_4 \in \{\top,\ \bar{\top}\}, \\ \bot & \text{if } F_4 \in \{\bot,\ \bar{\bot}\}, \\ R_+(\boldsymbol{t}) & \text{if } F_4 = R(\boldsymbol{t}), \\ R_-(\boldsymbol{t}) & \text{if } F_4 = \neg R(\boldsymbol{t}), \\ \neg R_+(\boldsymbol{t}) & \text{if } F_4 = \sim R(\boldsymbol{t}) \\ \neg R_-(\boldsymbol{t}) & \text{if } F_4 = \sim \neg R(\boldsymbol{t}), \\ (\lambda(G) \wedge \lambda(H)) & \text{if } F_4 \in \{G \wedge H,\ G \otimes H\}; \\ (\lambda(G) \vee \lambda(H)) & \text{if } F_4 \in \{G \vee H,\ G \oplus H\}; \\ (Qx)\lambda(F) & \text{for a formula } F \text{ with } Q \text{ as quantifier.} \end{cases}$$

Figure A.2: The extended mapping $\lambda : F_4 \mapsto F_2$ for $\langle \mathcal{A}_\rightarrow, \mathcal{L}_\rightarrow, \models_t \rangle$ considering formulas transformed by the convolution based Negation Normal Form Algorithm.

Mapping λ for convolution

For the reader interested in how formulas are mapped to classical first-order logic that are transformed by the convolution based Negation Normal Form Algorithm, we provide the respective mapping λ in Figure A.2. It is important to notice that the convolution translates to a negation in classical first-order logic and, therefore, our representation in classical first-order logic is not negation free anymore. This imposes the question whether the representation in a classical first-order logic causes some sets of formulas to explode. This question is not trivial and we have to distinguish two issues that may lead to explosion. The first is linked to the occurrence of the False Formula and the second is linked to the classical negation connective. We illustrate both issues with respect to a set $\mathcal{F} \subseteq \mathcal{L}_\rightarrow$ of our four-valued first-order logic $\langle \mathcal{A}_\rightarrow, \mathcal{L}_\rightarrow, \models_t \rangle$. We denote by $\mathcal{F}_\lambda$ the set of the formulas of $\mathcal{F}$ that are mapped to our classical first-order logic $\langle \mathcal{A}_{2\rightarrow}, \mathcal{L}_{2\rightarrow}, \models_2 \rangle$. Obviously, if one of the elements of $\mathcal{F}$ is the False Formula, then $\mathcal{F}_\lambda$ contains the False Formula and, thus, $\mathcal{F}_\lambda$ explodes. This fact is not new since in this case even $\mathcal{F}$ explodes. More attention deserves the second issue, i.e., the classical negation connective. If some formula of $\mathcal{F}$ translates to the form $\neg F \wedge F$, then $\mathcal{F}_\lambda$ explodes. However, this does not occur with formulas that are expressible in $\langle \mathcal{A}_\rightarrow, \mathcal{L}_\rightarrow, \models_t \rangle$. The reason for that is intuitively that our implication connective preserves full paraconsistency. In the following, we illustrate this issue by an example.

Example A.2
Let $F \rightarrow G$ be a formula of the four-valued first-order logic $\langle \mathcal{A}_\rightarrow, \mathcal{L}_\rightarrow, \models_t \rangle$, where F and G are atomic. λ maps $F \rightarrow G$ to $\neg F_+ \vee G_+$ which can be represented without negation as $F_+ \rightarrow G_+$. A negation of F or G just changes the atom to which we map in $\langle \mathcal{A}_{2\rightarrow}, \mathcal{L}_{2\rightarrow}, \models_2 \rangle$, i.e., one of F_+, F_-, G_+ or G_-. Thus, no classical negation is introduced. If we consider additionally the formula $\neg(F \rightarrow G)$, where F and G are atomic, we find that λ maps $\neg(F \rightarrow G)$

to $F_- \wedge G_-$ such that no classical negation is introduced. That is to say, we do not necessarily require for the classical negation connective to represent the formulas of $\langle \mathcal{A}_\rightarrow, \mathcal{L}_\rightarrow, \models_t \rangle$ in the classical first-order logic $\langle \mathcal{A}_{2\rightarrow}, \mathcal{L}_{2\rightarrow}, \models_2 \rangle$. Consequently, no inconsistencies can be introduced by mapping formulas of $\mathcal{F}$ to $\langle \mathcal{A}_{2\rightarrow}, \mathcal{L}_{2\rightarrow}, \models_2 \rangle$ and, thus, $\mathcal{F}_\lambda$ does not explode.

Example A.2 can be extended to arbitrary formulas. We omit doing so because the claim becomes obvious by proving Lemma 3.5.16 for the mapping λ in Figure A.2.

Proof of Lemma 3.5.16
We extend the proof that we gave on Page 32 by the necessary additionally induction steps. We restrict the proof to formulas in Negation Normal Form. This is possible without loss of generality because every formula of $\langle \mathcal{A}_\rightarrow, \mathcal{L}_\rightarrow, \models_t \rangle$ can be transformed to Negation Normal Form.

Only-if: Assume that I_4, σ is a model of a four-valued first-order logic formula F_4. We show that the corresponding interpretation I_2 of $\langle \mathcal{A}_{2\rightarrow}, \mathcal{L}_{2\rightarrow}, \models_2 \rangle$ and σ constitute a model of $\lambda(F_4)$.

Induction Basis: We add the cases for $\top$, $\bot$, $\overline{\top}$, $\underline{\bot}$.

7. If F_4 is of the form $\top$ or $\overline{\top}$, then $I_4, \sigma \models_t \top$ or $I_4, \sigma \models_t \overline{\top}$ for all I_4, σ. In this case, we find that $I_2, \sigma \models_2 \top$ for all I_2, σ and, therefore, I_2, σ is a model of $\lambda(F_4)$.

8. If F_4 is of the form $\bot$ or $\underline{\bot}$, then I_4, σ is not a model of F_4.

Induction Steps: We add the cases for $\sim$, $\otimes$, and $\oplus$.

9. If F_4 is of the form $\sim G$, then $I_4, \sigma \models_t \sim G$ and $I_4, \sigma \not\models_t G$. The induction hypothesis implies that $I_2, \sigma \not\models_2 \lambda(G)$ and, thus, $I_2, \sigma \models_2 \neg\lambda(G)$. Hence, I_2, σ is a model of $\lambda(F_4)$.

10. If F_4 is of the form $G \oplus H$, then $I_4, \sigma \models_t G \oplus H$, which in turn is the case if and only if

$$
\begin{aligned}
& I_4, \sigma \models_t \sim (\sim G \wedge \sim H) \\
\text{iff} \quad & I_4, \sigma \not\models_t \sim G \wedge \sim H \\
\text{iff} \quad & I_4, \sigma \not\models_t \sim G \text{ or } I_4, \sigma \not\models_t \sim H \\
\text{iff} \quad & I_4, \sigma \models_t G \text{ or } I_4, \sigma \models_t H \\
\text{iff} \quad & I_4, \sigma \models_t G \vee H.
\end{aligned}
$$

That is to say, I_4, σ is a model of $G \oplus H$ if and only if it is a model of $G \vee H$. The induction hypothesis implies that $I_2, \sigma \models_2 \lambda(G)$ or $I_2, \sigma \models_2 \lambda(H)$. Hence, $I_2, \sigma \models_2 \lambda(G \vee H)$ and consequently I_2, σ is a model of $\lambda(F_4)$.

11. If F_4 is of the form $G \otimes H$, then $I_4, \sigma \models_t G \otimes H$, which in turn is the case if and only if

$$
\begin{aligned}
& I_4, \sigma \models_t \sim (\sim G \vee \sim H) \\
\text{iff} \quad & I_4, \sigma \not\models_t \sim G \vee \sim H \\
\text{iff} \quad & I_4, \sigma \not\models_t \sim G \text{ and } I_4, \sigma \not\models_t \sim H \\
\text{iff} \quad & I_4, \sigma \models_t G \text{ and } I_4, \sigma \models_t H \\
\text{iff} \quad & I_4, \sigma \models_t G \wedge H.
\end{aligned}
$$

That is to say, I_4, σ is a model of $G \otimes H$ if and only if it is a model of $G \wedge H$. The induction hypothesis implies that $I_2, \sigma \models_2 \lambda(G)$ and $I_2, \sigma \models_2 \lambda(H)$. Hence, $I_2, \sigma \models_2 \lambda(G \wedge H)$ and consequently I_2, σ is a model of $\lambda(F_4)$.

If: Assume that an interpretation I_2, σ is a model of a formula F_2 in the classical first-order logic $\langle \mathcal{A}_{2\rightarrow}, \mathcal{L}_{2\rightarrow}, \models_2 \rangle$. We show that the corresponding interpretation I_4 and σ constitute a model of $\lambda^{-1}(F_2)$.

Induction Basis: We add the cases for $\top$ and $\bot$.

7. If F_2 is of the form $\top$, then $I_2, \sigma \models_2 \top$ for all I_2, σ. In this case, we find that $I_4, \sigma \models_t \top$ for all I_4, σ and, therefore, I_4, σ is a model of $\lambda^{-1}(F_2)$.

8. If F_2 is of the form $\bot$, then I_2, σ is not a model of F_2.

Induction Step: We add the cases for negation.

9. If F_2 is of the form $\neg G$, then $I_2, \sigma \models_2 \neg G$ and $I_2, \sigma \not\models_2 G$. The induction hypothesis implies that $I_4, \sigma \not\models_t \lambda^{-1}(G)$ and, thus, $I_4, \sigma \models_t \sim \lambda^{-1}(G)$. Hence, I_4, σ is a model of $\lambda^{-1}(F_2)$.

Bibliography

[AA94]   Ofer Arieli and Arnon Avron. Logical Bilattices and Inconsistent
           Data. In *Logic in Computer Science*, pages 468–476, 1994.

[AA95]   Ofer Arieli and Arnon Avron. A Bilattice-Based Approach to Re-
           cover Consistent Data From Inconsistent Knowledge-Bases. In *Pro-
           ceedings of the 4th Bar-Ilan Symposium on Foundations of Artificial
           Intelligence (BISFAI 95)*, pages 14–23. AAAI Press, 1995.

[AA96]   Ofer Arieli and Arnon Avron. Reasoning with Logical Bilattices.
           Journal of Logic, Language and Information, 5(1):25–63, 1996.

[AA97]   Ofer Arieli and Arnon Avron. Bilattices and Paraconsistency. In
           *Proceedings of the 1st World Conference on Paraconsistency (WCP
           97)*, 1997.

[AA98]   Ofer Arieli and Arnon Avron. The value of the four values. *Artificial
           Intelligence*, 102(1):97–141, 1998.

[AB90]   Alan R. Anderson and Nuel D. Belnap. *Entailment: the Logic of
           Relevance and Necessity*, volume 1. Princeton University Press,
           Princeton, New Jersey, second edition, 1990.

[AD01]   Ofer Arieli and Marc Denecker. Circumscriptive approaches to
           paraconsistent reasoning, 2001.

[AD02]   Ofer Arieli and Marc Denecker. Modeling Paraconsistent Reasoning
           by Classical Logic. In Thomas Eiter and Klaus-Dieter Schewe,
           editors, *FoIKS*, volume 2284 of *Lecture Notes in Computer Science*,
           pages 1–14. Springer, 2002.

[Ari98]   Ofer Arieli. Four-Valued Logics for Reasoning with Uncertainty in
           Prioritized Data. In *Proceedings of the 7th Conference on Infor-
           mation Processing and Management of Uncertainty in Knowledge-
           Based Systems (IPMU'98)*, pages 503–510, 1998.

[Ase66]   F. G. Asenjo. A calculus of antinomies. *Notre Dame Journal of
           Formal Logic*, 7(1):103–105, 1966.

[AT75]   F. G. Asenjo and J. Tamburino. Logic of Antinomies. *Notre Dame
           Journal of Formal Logic*, 16(1):17–44, 1975.

[Avr99]   Arnon Avron. On the Expressive Power of Three-Valued and Four-Valued Languages. *Journal of Logic and Computation*, 9(6):977–994, 1999.

[BBC03]   Salem Benferhat, Rania El Baida, and Frédéric Cuppens. A stratification-based approach for handling conflicts in access control. In *SACMAT*, pages 189–195. ACM, 2003.

[BCM+03]   Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[Bel75]   Nuel D. Belnap. A Useful Four-Valued Logic. In J. Michael Dunn and G. Epstein, editors, *Modern Uses of Multiple-Valued Logic*, pages 8–37. D. Reidel Publisher, 1975.

[Bel76]   Nuel D. Belnap. How A Computer Should Think. In G. Ryle, editor, *Contemporary Aspects of Philosophy*, pages 30–56. Oriel Press, 1976.

[Béz98]   Jean-Yves Béziau. Idempotent Full Paraconsistent Negations Are Not Algebraizable. *Notre Dame Journal of Formal Logic*, 39 (1)(1):135–139, 1998.

[Béz99]   Jean-Yves Béziau. The Future of Paraconsistent Logic. *Logical Journal Studies*, 2, 1999.

[Béz02]   Jean-Yves Béziau. $S5$ is a Paraconsistent Logic and so is Classical First-Order Logic. *Logical Journal Studies*, 9:301–309, 2002.

[BHS05a]   Leopold E. Bertossi, Anthony Hunter, and Torsten Schaub, editors. *Inconsistency Tolerance [result from a Dagstuhl seminar]*, volume 3300 of *Lecture Notes in Computer Science*. Springer, 2005.

[BHS05b]   Leopoldo E. Bertossi, Anthony Hunter, and Torsten Schaub. Introduction to Inconsistency Tolerance. In Bertossi et al. [BHS05a], pages 1–14.

[CMM05]   Sylvie Coste-Marquis and Pierre Marquis. On the Complexity of Paraconsistent Inference Relations. In Bertossi et al. [BHS05a], pages 151–190.

[CS96]   Marco Cadoli and Marco Schaerf. On the complexity of Entailment in Propositional Multivalued Logics. *Ann. Math. Artif. Intell.*, 18(1):29–50, 1996.

[dC74]   Newton C. A. da Costa. On the Theory of Inconsistent Formal System. *Notre Dame Journal of Formal Logic*, 15 (4):497–510, 1974.

[dCBB95]   Newton C. A. da Costa, Jean-Yves Béziau, and Otávio A. S. Bueno. Aspects of Paraconsistent Logics. *Bulletin of the Interest Group in Pure and Applied Logic*, 3(4):597–614, 1995.

[dCKB04]   Newton C. A. da Costa, Décio Krause, and Otávio A. S. Bueno. Paraconsistent Logics and Paraconsistency: Technical and Philosophical Developments, 2004.

[dCLLP05] Paulo Cesar G. da Costa, Kathryn B. Laskey, Kenneth J. Laskey, and Michael Pool, editors. *International Semantic Web Conference, ISWC 2005, Galway, Ireland, Workshop 3: Uncertainty Reasoning for the Semantic Web, 7 November 2005*, 2005.

[DR86]   J. Michael Dunn and Greg Restall. Relevance Logic. In D. M. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, volume 6, pages 1–128. D. Reidel Publisher, 1986.

[Fit90]   Melvin Fitting. Bilattices in Logic Programming. In G. Epstein, editor, *The 20th Int. Symp. on Multiple-Valued Logic*, pages 238–246. IEEE Press, 1990.

[Fit91a]   Melvin Fitting. Bilattices and the Semantics of Logic Programming. *Journal of Logic Programming*, 11(1&2):91–116, 1991.

[Fit91b]   Melvin Fitting. Kleene's Logic, Generalized. *Journal of Logic and Computation*, 1(6):797 – 810, 1991.

[Fit94]   Melvin Fitting. Kleene's Three Valued Logics and Their Children. *Fundamenta Informaticae*, 20(1/2/3):113–131, 1994.

[Fit95]   Melvin Fitting. Tableaus for Many-Valued Modal Logic. *Studia Logica*, 55(1):63–87, 1995.

[Fit96]   Melvin Fitting. *First-Order Logic and Automated Theorem Proving*. Springer, second edition, 1996.

[GHJV95]   Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns*. Addison-Wesley Professional, 1995.

[Gin88]   Metthew L. Ginsberg. Multivalued logics: a uniform approach to reasoning in artificial intelligence. *Computational Intelligence*, 4:265–316, 1988.

[Gin90]   Metthew L. Ginsberg. Bilattices and Modal Operators. In Rohit Parikh, editor, *Theoretical Aspects of Reasoning about Knowledge: Proceedings of the Third Conference (TARK 1990)*, pages 273–287, Los Altos, California, 1990. Morgan Kaufmann.

[HEPS03]   Masahiro Hori, Jérôme Euzenat, and Peter F. Patel-Schneider. World Wide Web Consortium – OWL Web Ontology Language XML Presentation Syntax. *http://www.w3.org/TR/owl-xmlsyntax*, 11 June, 2003.

[HMS04]   Ullrich Hustadt, Boris Motik, and Ulrike Sattler. Reasoning for De-
            scription Logics around $\mathcal{SHIQ}$ in a Resolution Framework. Tech-
            nical Report 3-8-04/04, FZI, Karlsruhe, Germany, June 2004.

[Höl00]   Steffen Hölldobler. Computational Logic – Working Material.
            http://www.computational-logic.org/∼sh/pub.html, 2000.

[Höl03]   Steffen Hölldobler. *Logik und Logikprogrammierung.* Synchron Ver-
            lag, Heidelberg, 3 edition, 2003.

[HS98]   Ian Horrocks and Ulrike Sattler. A Description Logic with Tran-
            sitive and Converse Roles and Role Hierarchies. In *Proceedings of
            the International Workshop on Description Logics*, Povo - Trento,
            Italy, 1998. IRST.

[HST00]   Ian Horrocks, Ulrike Sattler, and Stephan Tobies. Reasoning
            with Individuals for the Description Logic $\mathcal{SHIQ}$. In David A.
            McAllester, editor, *Proceedings of the 17th International Confer-
            ence on Automated Deduction (CADE-17)*, volume 1831 of *Lecture
            Notes in Computer Science.* Springer, 2000.

[Hun98]   Anthony Hunter. Paraconsistent Logics. In D. M. Gabbay, ed-
            itor, *Handbook of Defeasible Reasoning and Uncertainty Manage-
            ment Systems*, volume 2, pages 11–36. Kluwer Academic Publishers,
            1998.

[HV05]   Peter Haase and Johanna Völker. Ontology Learning and Reason-
            ing – Dealing with Uncertainty and Inconsistency. In da Costa et al.
            [dCLLP05], pages 45–55.

[HvHtT05]   Zhisheng Huang, Frank van Harmelen, and Annette ten Teije. Rea-
            soning with Inconsistent Ontologies. In K. Verbeeck, K. Tuyls,
            A. Nowé, B. Manderick, and B. Kuijpers, editors, *BNAIC*, pages
            349–350. Koninklijke Vlaamse Academie van Belie voor Weten-
            schappen en Kunsten, 2005.

[IH00]   Stephan Tobies Ian Horrocks, Ulrike Sattler. Practical Reasoning
            for Very Expressive Description Logics. *Logic Journal of the IGPL*,
            8(3):239–264, 2000.

[Jas69]   Stanislaw Jaskowski. Propositional Calculus for Contradictory De-
            ductive Systems. *Studia Logica*, 14:143–157, 1969.

[Kno02]   Kirk Knoernschild. *Java Design.* Addison-Wesley, 2002.

[ML06]   Yue Ma and Zuoquan Lin. Inferring with Inconsistent OWL DL
            Ontology: a Multi-valued Approach, International Conference on
            Semantics of a Networked World, ICSNW 2006, Munich, Germany,
            March 30, 2006. In *ICSNW*. Springer, to appear 2006.

[MS96]   Carlo Meghini and Umberto Straccia. A Relevance Terminolog-
            ical Logic for Information Retrieval. In H.-P. Frei, D. Harman,
            P. Schäuble, and R. Wilkinson, editors, *SIGIR*, pages 197–205.
            ACM, 1996.

[Mus99]   Reinhard Muskens. On Partial and Paraconsistent Logics. *Notre
            Dame Journal of Formal Logic*, 40(3):352–374, 1999.

[Pri00]   Graham Priest. Inconsistent Models of Arithmetic Part II: the
            General Case. *Journal of Symbolic Logic*, 65(4):1519–1529, 2000.

[Pri02]   Graham Priest. Paraconsistent Logic. In D. M. Gabbay and
            F. Guenthner, editors, *Handbook of Philosophical Logic*, volume 6,
            pages 287–393. Kluwer Academic Publishers, second edition, 2002.

[PS88]   Peter F. Patel-Schneider. Adding Number Restrictions to a Four-
            Valued Terminological Logic. In *AAAI*, pages 485–490, 1988.

[PS89]   Peter F. Patel-Schneider. A Four-Valued Semantics for Termino-
            logical Logics. *Artificial Intelligence*, 38(3):319–351, 1989.

[PT04]   Graham Priest and Koji Tanaka. Paraconsistent Logic. In Ed-
            ward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*.
            Winter 2004.

[Res03]   Greg Restall. Paraconsistency Everywhere. *Notre Dame Journal
            of Formal Logic*, 43 (3):147–156, 2003.

[RR98]   Odinaldo Rodrigues and Alessandra Russo. A Translation Method
            for Belnap Logic, September 1998.

[Rue96]   Paul Ruet. Complete sets of connectives and complete se-
            quent calculus for belnap's logic. Technical report, LIENS-96-28,
            Département de Mathématiques et Infromatique, Ecole Normale
            Superieure, 1996.

[SBB⁺05]   Sebastian Schaffert, François Bry, Philippe Besnard, Hendrik
            Decker, Stefan Decker, Carlos F. Enguix, and Andreas Herzig.
            Paraconsistent Reasoning for the Semantic Web. In da Costa et al.
            [dCLLP05], pages 104–105.

[SC03]   Stefan Schlobach and Ronald Cornet. Non-Standard Reasoning
            Services for the Debugging of Description Logic Terminologies. In
            G. Gottlob and T. Walsh, editors, *IJCAI*, pages 355–362. Morgan
            Kaufmann, 2003.

[Sch05]   Stefan Schlobach. Debugging and Semantic Clarification by Pin-
            pointing. In A. Gómez-Pérez and J. Euzenat, editors, *ESWC*, vol-
            ume 3532 of *Lecture Notes in Computer Science*, pages 226–240.
            Springer, 2005.

[Str97]   Umberto Straccia.   A Sequent Calculus for Reasoning in Four-
            Valued Description Logics. In D. Galmiche, editor, *TABLEAUX*,
            volume 1227 of *Lecture Notes in Computer Science*, pages 343–357.
            Springer, 1997.

[Str99]   Umberto Straccia. *Foundations of a Logic based approach to Multi-
            media Document Retrieval*. PhD thesis, Department of Computer
            Science, University of Dortmund, June 1999.

[SWM04]   Michael K. Smith, Chris Welty, and Deborah L. McGuinness. World
            Wide Web Consortium – OWL Web Ontology Language Guide.
            http://www.w3.org/TR/2004/REC-owl-guide-20040210, 10 Febru-
            ary, 2004.

[Tan03]   Koji Tanaka. Three Schools of Paraconsistency. *The Australasian
            Journal of Logic*, 1:28–42, 2003.

[Tso02]   Alexis Tsoukiàs. A first-order, four valued, weakly paraconsistent
            logic and its relation to rough sets semantics. *Foundations of Com-
            puting and Decision Sciences*, 12:85–108, 2002.

[Vil02]   Jørgen Villadsen.   Paraconsistent Knowledge Bases and Many-
            Valued Logic. In *BalticDB&IS*, pages 77–90, 2002.

[VVS05]   Johanna Völker, Denny Vrandecic, and York Sure. Automatic Eval-
            uation of Ontologies (AEON). In Y. Gil, E. Motta, V. R. Ben-
            jamins, and M. A. Musen, editors, *International Semantic Web
            Conference*, volume 3729 of *Lecture Notes in Computer Science*,
            pages 716–731. Springer, 2005.

[Wit81]   Ludwig Wittgenstein.   *Philosophische Bemerkungen*.   Suhrkamp
            Taschenbuch Verlag, Frankfurt am Main, 1 edition, 1981.

[WOS03]   Heinrich Wansing, Sergei P. Odintsov, and Yaroslav Shramko.
            Inconsistency-tolerant Description Logic: Motivation and Basic
            Systems. In V. Hendricks and J. Malinowski, editors, *Trends in
            Logic: 50 Years of Studia Logica*, pages 301–335. Kluwer Academic
            Publishers, 2003.

Index

Statement of Academic Honesty

Hereby I declare that this is my work and that I did not use any other sources than the ones cited.

Selbstständigkeitserklärung

Hiermit erkläre ich, daß die vorliegende Arbeit von mir selbst verfaßt wurde, und ich keine anderen als die angegebenen Quellen benutzt habe.

Andreas Lang
Technische Universität Dresden,
März, 2006