Universidade Nova de Lisboa
Faculdade de Ciências e Tecnologia
Departamento de Informática

Combining Open and Closed World Reasoning for the Semantic Web

Matthias Knorr

Dissertação para obtenção do Grau de Doutor em Informática

Orientador: Prof. Doutor José Júlio Alferes
Co-orientador: Prof. Doutor Pascal Hitzler

Júri

Presidente: Doutora Elvira Júlia Conceição Matias Coimbra

Arguente: Doutor Riccardo Rosati
Arguente: Doutor Gerhard Brewka

Vogal: Doutor Carlos Manuel Costa Lourenço Caleiro
Vogal: Doutora Helena Sofia Andrade Nunes Pereira Pinto
Vogal: Doutor João Alexandre Carvalho Pinheiro Leite

Maio 2011

Combining Open and Closed World Reasoning for the Semantic Web. Copyright by Matthias Knorr, FCT/UNL, UNL, 2011.
A Faculdade de Ciências e Tecnologia e a Universidade Nova de Lisboa têm o direito, perpétuo e sem limites geográficos, de arquivar e publicar esta dissertação através de exemplares impressos reproduzidos em papel ou de forma digital, ou por qualquer outro meio conhecido ou que venha inventado, e de divulgar através de repositórios científicos e de admitir a sua cópia e distribuição com objectivos educacionais ou de investigação, não comerciais, desde que seja dado crédito ao autor e editor.

Acknowledgements

First of all, I want to thank José Júlio Alferes and Pascal Hitzler for supervising this work. This thesis would not have been possible without their incentives, their guidance and support during the last years, their availability for discussions in person or by mail, and their patience when reading and discussing preliminary versions of this thesis or prior scientific publications. They introduced me to scientific work and the scientific community and I am very grateful for that. I want to thank particularly José Júlio Alferes for being available and taking care of any bureaucratic obstacle that required some insight or language capability I did not have, including the help on the abstract in Portuguese.

Part of the work presented in this thesis was developed in collaboration with Terrance Swift. I am thankful for the joint discussions, his detailed introduction into the subtleties of SLG resolution with tabling, and, not the least, his help with software issues, such as creating the improved figure of the derivation forest. I also want to thank all my colleagues from CENTRIA, and in particular the participants of the KRR seminars (and its predecessors, such as WALK) for the interesting presentations and discussions (during the seminars but not limited to that). They created a pleasant working environment, which besides their comments on this work most certainly had an additional positive impact on it. I am indebted to all those anonymous reviewers whose comments helped improve papers in which part of the developed work was presented.

I want to thank for the working space and the institutional support of this thesis by the Departamento de Informática of Faculadade para a Ciência e a Tecnologia, Universidade Nova de Lisboa. This thesis was supported by

FCT (Fundação para a Ciência e a Tecnologia) under the grant contract SFRH/BD/28745/2006.

A special thanks goes to my family for all their support and care during my entire life and the time of the PhD in spite of the distance. My final thanks for being with me go to Márcia.

Abstract

One important problem in the ongoing standardization of knowledge representation languages for the Semantic Web is combining open world ontology languages, such as the OWL-based ones, and closed world rule-based languages. The main difficulty of such a combination is that both formalisms are quite orthogonal w.r.t. expressiveness and how decidability is achieved. Combining non-monotonic rules and ontologies is thus a challenging task that requires careful balancing between expressiveness of the knowledge representation language and the computational complexity of reasoning.

In this thesis, we will argue in favor of a combination of ontologies and non-monotonic rules that tightly integrates the two formalisms involved, that has a computational complexity that is as low as possible, and that allows us to query for information instead of calculating the whole model. As our starting point we choose the mature approach of hybrid MKNF knowledge bases, which is based on an adaptation of the Stable Model Semantics to knowledge bases consisting of ontology axioms and rules. We extend the two-valued framework of MKNF logics to a three-valued logics, and we propose a well-founded semantics for non-disjunctive hybrid MKNF knowledge bases. This new semantics promises to provide better efficiency of reasoning, and it is faithful w.r.t. the original two-valued MKNF semantics and compatible with both the OWL-based semantics and the traditional Well-Founded Semantics for logic programs. We provide an algorithm based on operators to compute the unique model, and we extend SLG resolution with tabling to a general framework that allows us to query a combination of non-monotonic rules and any given ontology language. Finally, we investigate concrete instances of that procedure w.r.t. three tractable ontology languages, namely the three description logics underlying the OWL 2 profiles.

Key words: Semantic Web, Non-monotonic Reasoning, Description Logics, Logic Programming

Resumo

A combinação de linguagens de ontologias baseadas na assunção de mundo aberto (como a linguagem OWL), com linguagens de regras baseadas na assunção de mundo fechado, é um dos assuntos importantes para a estandardização de linguagens de representação de conhecimento na "Semantic Web". A maior dificuldade nesta combinação reside no factos de ambos os formalismos serem ortogonais, tanto no que diz respeito à sua expressividade, como no que respeita à forma como em ambos os casos se restringem as linguagens para obter decidibilidade. Assim, a combinação de ontologias e regras não-monotónicas constitui um desafio que requer um equilíbrio cuidadoso entre a expressividade da linguagem de representação de conhecimento e a complexidade da computação do raciocínio.

Esta dissertação defende uma forma de combinação de ontologias e regras não-monotónicas com um grande nível de integração, e mantendo uma complexidade computacional tão baixa quanto possível. Além disso, defende-se também a importância, para aplicações à "Semantic Web", de procedimentos eficientes para resposta a perguntas que não exijam o cálculo de modelos completos, e apresentam-se procedimentos que cumprem com esse requisito.

Como ponto de partida para a proposta aqui apresentada, foi escolhida a lógica híbrida MKNF, a qual se baseia numa adaptação dos modelos estáveis para bases de conhecimento que consistem em axiomas ontológicos e regras. A lógica MKNF original, a dois valores, é estendida para uma lógica a três valores e propõe-se uma semântica bem fundada para bases do conhecimento híbridas MKNF não disjunctivas. Esta nova semântica promete fornecer uma maior eficiência de raciocínio, é fiel relativamente à semântica original MKNF a dois valores, e é compatível quer com a semântica do OWL quer com a semântica bem fundada tradicional para programas em lógica. É ainda definido um algoritmo baseado em operadores para computar o modelo bem fundado completo de uma base de conhecimento, e um procedimento geral de resolução, que estende o SLG com tabelação, e que

permite responder a perguntas sobre bases de conhecimento com regras não-monotónicas e ontologias descritas por uma qualquer lógica de descrição. Finalmente, são investigados casos concretos deste procedimento, respeitantes às três lógicas de descrição que se encontram na base dos perfis do OWL 2, mostrando que nesses casos a introdução de regras não-monotónicas não faz aumentar o nível de complexidade.

Palavras-chave: Semantic Web, Raciocínio Não-monotónico, Lógicas de Descrição, Programação em Lógica

Contents

List of Figures **xiii**

I Combining Rules and Ontologies **1**

1 Introduction **3**

1.1 OWA – Description Logics 4
1.2 CWA – Logic Programming Rules 7
1.3 Open vs. Closed World Reasoning 9
1.4 The Problem: Combining Rules and Ontologies efficiently 12
1.5 Existing Combinations of Rules and Ontologies 15
1.6 A Novel Approach 18
1.7 Contributions 20
1.8 Outline 21

2 Rules in Logic Programming **25**

2.1 Fixed Point Semantics for Logic Programs 26
2.2 Computational Complexity 28
2.3 Terminology of Logic Programs 30
2.4 Answer Set Semantics 33
2.5 Stable Model Semantics 34
2.6 Well-founded Semantics 36

3 $\mathcal{SROIQ}$ – An Expressive Description Logic **41**

3.1 Syntax of $\mathcal{SROIQ}$ 42
3.2 Semantics of $\mathcal{SROIQ}$ 47

3.3 Tractable Fragments of $\mathcal{SROIQ}$ 51

4 MKNF Logics and Hybrid MKNF Knowledge Bases 57

4.1 Syntax of MKNF Logics 58

4.2 Semantics of MKNF Logics 59

4.3 Standard Names Assumption 60

4.4 Hybrid MKNF$^+$ Knowledge Bases 62

4.5 Semantic Properties of Hybrid MKNF$^+$ 63

4.6 Decidability for Hybrid MKNF 65

4.7 Hybrid MKNF with Normal Rules 68

II A Well-founded Semantics for Hybrid MKNF Knowledge Bases 73

5 A Three-valued MKNF Semantics 75

5.1 Evaluation in MKNF Structures 76

5.2 Three-valued MKNF Models 79

5.3 General Properties of three-valued MKNF 82

5.4 A Well-Founded MKNF Model 83

5.5 Relation with the Two-valued MKNF Semantics 85

6 Alternating Fixpoint for the Well-founded MKNF Model 87

6.1 Partitions of Modal Atoms 88

6.2 Computation of the Alternating Fixpoint 93

6.3 An Alternative Characterization based on Unfounded Sets 105

6.4 The Well-Founded MKNF Model and Related Properties 111

7 Comparison to Related Approaches 121

7.1 Two-valued MKNF semantics 123

7.2 Combinations with First-Order Rules 125

7.3 Ontologies and Non-Monotonic Rules 131

7.4 Combinations based on the Well-Founded Semantics 140

III Querying Hybrid MKNF Knowledge Bases **147**

8 SLG($\mathcal{O}$)– A General Querying Procedure **149**

8.1 Alternative Bottom-Up Iteration 150

8.2 Top-Down Queries with SLGO 159

8.3 Properties of **SLG($\mathcal{O}$)** 169

9 Querying Tractable Hybrid MKNF Knowledge Bases **179**

9.1 An Oracle for $\mathcal{REL}$ 180

9.2 An Oracle for $DL\text{-}Lite_{\mathcal{R}}$ 188

9.3 An Oracle for DLP 196

10 Conclusions **201**

10.1 Accomplishments 201

10.2 Future Work 203

Index **206**

References **211**

List of Figures

3.1 Semantics of role and concept expressions in $\mathcal{SROIQ}$ for an interpretation $\mathcal{I}$ with domain $\Delta^{\mathcal{I}}$ 47
3.2 Semantics of $\mathcal{SROIQ}$ axioms for an interpretation $\mathcal{I}$ with domain $\Delta^{\mathcal{I}}$. 48
3.3 Syntax and semantics of $\mathcal{REL}$. 52
3.4 Syntactic restrictions on class expressions in $DL\text{-}Lite_{\mathcal{R}}$ where D is any allowed concept expression and $R \in \mathcal{R}$. 53
3.5 Syntactic restrictions on class expressions in DLP where C is any allowed concept expression and $R \in \mathcal{R}$. 54

4.1 Data complexity of instance checking in Admissible MKNF KBs 68

7.1 Comparison of combinations of ontologies with monotonic rules. 125
7.2 Comparison of combinations of ontologies with non-monotonic rules based on stable models. 132
7.3 Comparison of combinations of ontologies with non-monotonic rules based on well-founded semantics. 140

8.1 Final Forest for the query `discount(Bill)` to $\mathcal{K}$ 169

9.1 Algorithm Consistent 190
9.2 Algorithm $DL\text{-}Lite_{\mathcal{R}}$ Oracle 199

LIST OF FIGURES

Part I

Combining Rules and Ontologies

1

Introduction

Knowledge Representation and Reasoning (KRR) [Hayes, 1979; Levesque, 1984] is of major importance in the field of Artificial Intelligence[1] (AI). The general idea of KRR is to store knowledge about a domain of interest in so-called knowledge bases in a way that allows automated systems to access and use it, and that, even more importantly, allows the derivation of knowledge only implicitly present in the knowledge base (KB) under consideration.

In the last decade, the Semantic Web [Berners-Lee et al., May 2001] has become a major source of inspiration for KRR. The underlying idea of the Semantic Web is to augment the information of web pages with data that is machine-processable. In particular, KRR techniques are intended to be used to enhance data in the World Wide Web with knowledge bases, making this data available for processing by intelligent systems. The research is driven by the ongoing standardization process of the World Wide Web Consortium (W3C), which aims at bringing the Web to its full potential.[2] As such, Semantic Web has become a mature field of research, and industrial applications of Semantic Web technologies are on the way. Semantic Web is a topic that is clearly here to stay.

Nowadays, among the most prominent KRR formalisms applied in the Semantic Web are the ontology languages that are based on description logics. However, we believe that these ontology languages are not adequate for several application areas within the Semantic Web and one of the important problems [Shadbolt et al., 2006] is

[1] See, e.g., [Russell and Norvig, 2010] for an extensive introduction into AI and the area of KRR.
[2] `http://www.w3c.org`

the inability of expressing some form of closed-world modeling, such as the one provided by non-monotonic negation in Logic Programming. In fact, over the last few years a significant amount of scientific work on combining ontologies and some form of rule language has been presented (see, e.g., [Drabent et al., 2009; Eiter et al., 2008a; Hitzler and Parsia, 2009; Krisnadhi et al., 2011a] for a brief survey). This combination is of particular interest since the two formalisms provide quite different means for expressing knowledge and a combination of the two yields a richer KRR formalism.

Our thesis is that combining Description Logics underlying the ontology languages and non-monotonic rules of Logic Programming is of major benefit for the expressiveness of KRR formalisms in the Semantic Web, and that reasoning in such a combination can be achieved in an efficient way. This includes the possibility of querying for particular information in the knowledge base, thus considering only the information relevant for the query instead of computing/constructing a model for the entire knowledge base.

In the sequel, after introducing the two formalisms that realize open and closed world reasoning respectively, namely Description Logics (Section 1.1) and rules of logic programs (Section 1.2), we motivate why KRR formalisms combining open and closed world reasoning are sometimes preferable over fragments of classical first-order logics, such as description logics, and we present application scenarios illustrating the requirement for that combination (Section 1.3). We discuss why combining two such formalisms is a nontrivial task (Section 1.4). Then, we recall the already existing approaches of such combinations (Section 1.5), and we argue in favor of an approach that is as expressive, as general, and as robust as possible while its computational complexity remains as low as possible (Section 1.6). This motivates our work whose main contributions we present subsequently (Section 1.7) before we outline the rest of the thesis (Section 1.8).

1.1 OWA – Description Logics

Open world reasoning is based on the Open World Assumption (OWA). This means that (negative) conclusions drawn from a knowledge base must be based on information explicitly present in it. The most prominent highly expressive KRR approach employed in Semantic Web research is based on the Web Ontology Language OWL [Patel-Schneider et al., 2004], respectively the revised language OWL 2 [Hitzler et al.,

2009a]. OWL 2 is the new recommended standard by the W3C for modeling Semantic Web knowledge bases (commonly known as *ontologies*). Such ontology languages are based on Description Logics (DLs) [Baader et al., 2007a; Hitzler et al., 2009b], e.g., OWL is based on the description logic $\mathcal{SHOIN}(D)$, and OWL 2 is based on $\mathcal{SROIQ}(D)$. Description Logics bear a first-order predicate logic semantics, they are monotonic and adhere to the OWA.

Description Logics have evolved[1] from semantic networks [Quillian, 1967] and frames [Minsky, 1981]. Both these early KRR formalisms are based on the notions of classes of individuals and relations between these classes, and the main motivation for their development was the intention of representing taxonomic knowledge, i.e., relations between classes of individuals. The problem of semantic networks and frames is the lack of formal semantics – a relation between two classes could mean that either there is some relation between the individuals of these classes, or that the relation is true for all individuals of these related classes, or even that the relation is a default relation that holds until contradictory knowledge is available explicitly. The consequence is that a lot of the early systems based on such so-called network-based structures [Lehmann, 1992] behave differently, even though they appear to be almost identical.

In [Hayes, 1979], it was realized that frames could basically be given a semantics by relying on first-order logics: sets of individuals can be represented by unary predicates, and relations between such sets can be represented by binary predicates. By means of certain boolean constructors, we can express non-trivial sets of individuals, such as the set of all persons that have at least two children that are all female:

$$\texttt{C} \equiv (\geqslant 2\,\texttt{HasChild}) \sqcap (\forall\,\texttt{HasChild.Female}) \tag{1.1}$$

In this example, `Female` is a unary predicate, `HasChild` a binary one, $\sqcap$ represents intersection of sets of individuals, $\geqslant 2\,$`HasChild` specifies all individuals that are related to at least two other individuals via `HasChild`, and $\forall\,$`HasChild.Female` defines a class of all those individuals that relate via `HasChild` only to individuals that belong to the class `Female`. Such a class description can be translated straightforwardly into first-order logic (see, e.g., [Baader et al., 2007a]) and its semantics can be applied to the result of the translation. For example, the complex concept `C` above can be translated

[1] The historic overview is inspired by [Baader et al., 2007a], which presents further details.

into first-order logic with equality.

$$\begin{aligned}\phi_{\mathtt{C(x)}} =&(\exists \mathtt{y}, \mathtt{z} : \mathtt{HasChild}(\mathtt{x}, \mathtt{y}) \wedge \mathtt{HasChild}(\mathtt{x}, \mathtt{z}) \wedge (\mathtt{y} \neq \mathtt{z})) \\ &\wedge (\forall \mathtt{y} : \mathtt{HasChild}(\mathtt{x}, \mathtt{y}) \rightarrow \mathtt{Female}(\mathtt{y}))\end{aligned} \tag{1.2}$$

However, semantic networks and frames do not require the full expressiveness of first-order logic. It suffices to use fragments of it [Brachman and Levesque, 1985], and the intended interpretations for the relations in each of those network-based structures can be represented by different boolean constructors resulting in different fragments of first-order logic. As a consequence, it was recognized that reasoning in such structure-based representations could be achieved by specialized reasoners without relying on full first-order theorem provers, and that differing domains of interest lead to different fragments of first-order logic, and these result in computational problems of differing complexity.

The first system moving from semantic networks to such a formal semantics is KL-ONE [Brachman and Schmolze, 1985]. It introduced many of the notions used in Description Logics and the examination of KL-ONE and similar systems was the starting point for the description logic systems. In particular, these systems showed the importance of the trade-off between the expressiveness of a DL language and the complexity of its reasoning capabilities. CLASSIC [Brachman et al., 1991] permits only a limited set of constructors such that the computation is efficient and complete. Other approaches, such as LOOM [MacGregor and Bates, 1987] and BACK [Nebel and von Luck, 1988], were much more expressive but incomplete, in the sense that the systems were not able to detect all answers that are logically implied by the respective language fragment. This can be understood as a result of the fact that the underlying logical fragment is undecidable. Further investigations revealed that the source of incompleteness in such systems were certain combinations of constructs in the language. This led to the development of systems, such as KRIS [Baader and Hollunder, 1991], that were less efficient but expressive and complete.

The need for expressive and decidable DL languages has driven the development of more advanced systems, and tableaux-based algorithms were an important means for that. The most prominent modern approaches based on tableaux algorithms are FACT++ [Tsarkov and Horrocks, 2006], Pellet [Sirin et al., 2007], and RACER [Haarslev et al., 2011]. Alternative approaches are, e.g., KAON2 [Motik and Sattler, 2006], which is based on resolution, the hyper-tableaux system HermiT [Motik et al., 2009c],

approaches based on type elimination, such as [Rudolph et al., 2008a,b], and so-called consequence-based approaches [Kazakov, 2009].

Many of the before-mentioned systems were developed for some specific application and these applications of DL systems have been, and are being used in software engineering, configuration of systems, medicine, digital libraries, and Web-based information systems in the Semantic Web, but also planning, data mining, natural language processing, and database management (see, e.g., [Baader et al., 2007a] for details).

1.2 CWA – Logic Programming Rules

Closed world reasoning relies on the Closed World Assumption (CWA), under which all non-provable expressions are assumed to be false. This is used in formalisms, such as default reasoning [Reiter, 1980] or circumscription [McCarthy, 1980], but most famously as default negation in non-monotonic rules of Logic Programming.

Knowledge representation and reasoning formalisms based on rules themselves have a very long tradition. In fact, one can easily interpret Aristotle's syllogisms as rules since the term rule, as such, may refer to any statement that ensures that some premise implies a certain conclusion. Therefore, there is a large variety of formalisms based on rules and the ones most commonly used are rules in Logic Programming, association rules in data bases, and production rules in business rule systems. Among these, rules in Logic Programming are of particular interest since they admit the usage of the so-called default negation, an operator that allows non-monotonic reasoning in rules of logic programs. This distinctive feature, which is not expressible in first-order logic, makes logic program rules an appropriate means for closed world reasoning since it permits to model defaults and exceptions, and that is why we focus in this thesis on rules in logic programs.

Logic Programming (LP) [Apt and Bol, 1994; Baral and Gelfond, 1994; Lloyd, 1987] is a field that started around the same time as the first semantic network approaches (see, e.g., [Kowalski, 1988] for one account of the early days of LP). In opposite to the procedural paradigm of imperative programming languages, LP represents the declarative programming paradigm. The idea is to simply encode the knowledge and the problem of interest in a program, and to let the computer solve the problem without specifying a sequence in which the program has to be executed. Thus, the program

can be understood as a description of the problem. One may, e.g., represent that one's uncle is the brother of one's parent (where, as usual in LP notation, the variables are implicitly all-quantified):

$$\texttt{uncle}(\texttt{x},\texttt{z}) \leftarrow \texttt{brother}(\texttt{x},\texttt{y}),\texttt{parent}(\texttt{y},\texttt{z}) \tag{1.3}$$

This is obviously easy to understand as a plain representation of the essential relation. In a procedural definition we would have to devise how we retrieve the instances of the relations `brother` and `parent` and how we link them to obtain the proper `uncle` relation.

LP is closely related to the language of PROLOG [Colmerauer and Roussel, 1996], which was introduced in the early seventies. Given a logic program including the rule above, one can query the program for the truth of the uncle relation for two concrete given individuals or request some or even all instances of the uncle relation that are stored in the program. This relation between the declarative paradigm of LP and PROLOG systems was not immediately established but the results of theoretical research led ultimately to systems that were more advanced, more expressive, and closer to the declarative paradigm.

The first formal semantics for LP was the least model semantics [van Emden and Kowalski, 1976] for definite programs, i.e., programs whose rules consist of a conjunction of atomic formulas in the premise and only one conclusion. The obtained result states that there is a unique model for every definite program and was soon widely accepted. However, default negation (or negation as failure) in the premise of rules was not so easy to incorporate. The underlying Closed World Assumption [Reiter, 1978] aims at deriving some default negated information only if it is not possible to prove the contrary. This also means that the addition of new information may alter previously drawn conclusions. Hence, such normal logic programs are non-monotonic, and thus default negation is not embeddable into monotonic first-order logics.

Clark's completion [Clark, 1978] and the three-valued Fitting semantics [Fitting, 1985] provided a semantics to such programs, but sometimes these semantics yield counterintuitive results, such as a too weak notion of derivability or lack of semantics in some cases. These were eventually overcome by the two semantics that are nowadays considered the standard semantics for normal logic programs: the Stable Model Semantics (SMS) [Gelfond and Lifschitz, 1988] and the Well-Founded Semantics (WFS)

[van Gelder et al., 1991]. Both semantics are defined in terms of fixpoints of operators and in fact quite closely related [van Gelder, 1989]. Stable model semantics grew into the paradigm of Answer Set Programming (ASP) [Gelfond and Lifschitz, 1991] and corresponding systems, such as DLV [Leone et al., 2006], Smodels [Niemelä and Simons, 1997], or Clasp [Gebser et al., 2007], can be considered more declarative than PROLOG: e.g., in a standard PROLOG program the order in which the rules appear in it is of importance (and thus at least partially procedural), while ASP does not require that. The reason for this difference is that PROLOG systems answer queries, for which to some extent a search order has to be specified, while ASP systems simply compute models. In case of the well-founded semantics, XSB[1] can be considered the standard system providing the functionality of a PROLOG system but in a significantly improved way (more efficient and complete). All these modern systems are also more expressive, since they allow, e.g., the usage of classical negation in rules, or disjunctions as conclusions in case of ASP systems.

Logic programming systems have been used early on for planning and natural language processing. The applicability has broadened since, in particular due to one of the extensions of LP – Constraint Logic Programming [Rossi et al., 2006], to a variety of fields, such as civil and mechanical engineering, circuit verification, control of autonomous agents, automated timetabling, air traffic control, bioinformatics, and also Web-based information systems (see, e.g., Rule Interchange Format (RIF-Core [Boley et al., 2010] and RIF-BLD [Boley and Kifer, 2010]) or systems based on F-Logic [Kifer et al., 1995]).

1.3 Open vs. Closed World Reasoning

Open world reasoning and closed world reasoning rely on inherently different assumptions. The decision when to apply which assumption in KRR depends on the reasoning tasks and the application in mind.

In case of ontology languages, such as OWL, the OWA, and thus open world reasoning, is applied. The decision to rely on the OWA appears to be a natural one in light of the envisioned applications related to the World Wide Web: the absence of a piece of knowledge should not generally be taken as an indication that this piece

[1] `http://xsb.sourceforge.net`

of knowledge is false. However, there are also application scenarios where the CWA, or at least the partial closure of the knowledge base, is a more natural choice. Such scenarios can occur, e.g., if ontology-based reasoning is done in conjunction with data stored in a database. Database data is usually considered to be complete, and so statements not in the database should be taken as false. Alternatively, we might want to model default information, exceptions, or constraints, and in this case logic programs of non-monotonic rules are a natural choice.

As a concrete example where a combination of OWA and CWA is desired, consider the large case study described in [Patel et al., 2007], containing millions of assertions about matching patient records with clinical trials criteria. In this clinical domain, open world reasoning is needed in radiology and laboratory data. For example, unless a lab test asserts a negative finding, no arbitrary assumptions about the results of the test can be made. That is, we can only be certain that some patient does not have a specific kind of cancer if the corresponding test has a negative result. However, as observed in [Patel et al., 2007], the closed world assumption can and should be used with data about medical treatment to infer that a patient is not on a medication unless otherwise stated. The work of [Patel et al., 2007] applies only open world reasoning but claims that the usage of closed world reasoning in data about medical treatment would be highly desirable and that the combination of OWA and CWA is an open problem in their work. Similar situations occur, e.g., in matchmaking using Semantic Web Services (cf. [Grimm and Hitzler, 2008]), and in other scenarios in the medical domain.

In fact, life sciences, including medicine, is a prominently studied application area for OWL. Several large-scale ontologies have been developed in this area that are being used in practice, such as GALEN[1] and SNOMED.[2] These ontologies provide unified medical terminologies for the management and exchange of clinical information. The knowledge bases typically consist of information about anatomy, diseases, procedures, drugs, etc., and their applications range from medical record management to diagnostics support. SNOMED is used, for example, in the case study on matching patient records with clinical trials criteria described above. All of these applications use ontology reasoning based on the OWA. But it is not difficult to foresee situations in these domains that would benefit from local closed world reasoning. Consider, for example, that such a

[1] http://www.opengalen.org/

[2] http://www.ihtsdo.org/snomed-ct/

medical knowledge base is used to decide whether a certain anesthetic should be applied before surgery, depending on whether the patient is allergic to the anesthetic or not. This information might not be available, and it should be modeled using the CWA: in an emergency situation, unless we know explicitly about an allergy, we assume that the patient is not allergic, and we apply the anesthetic. Other examples can be found if we were to model exceptions in anatomical terminology; e.g., the existence of persons whose heart is actually on the right-hand side. Exception modeling is not directly possible in classical first-order logic (this is a problem usually known in Artificial Intelligence as the *specification problem*) and so also not possible in OWL using only the OWA.

Another application area is intelligent decision automation that can be used in telecommunications, health care, and financial services. One such system is AIDA[1] that is, e.g., applied in customer relation management in telecommunications. The conceptual idea is to store all the events related to customers (owned products, bills, previous requests, and so on) so that in case of a client's call the upcoming issue can literally be anticipated instead of starting the dialog with the client from zero. Events are stored in an ontology and the decision automation is realized in business rules. As an example for such a rule, a client who has an open bill payment is likely to phone to discuss that issue. Another example for such a rule is a client that recently bought a new device. It can reasonably be assumed that the client is calling because of some technical problem and the problem could be even more specifically targeted depending on whether the device is entirely new or just a substitute for one that the client already was using successfully. In such a scenario, in general the OWA is applied, since a priori the absence of some information does not allow us to draw any conclusions. But, e.g., the history of interactions of one client with the company is fully known, so CWA can be applied locally when reasoning over client specific information, such as the payment history or owned products. The system manages billions of events and, in a concrete example, the call center costs are reducible up to 40 percent. However, since the rules are incorporated in an ad-hoc manner, as stated in communications, it would be of importance to establish a formal basis for such systems making the ideas applicable to other areas as well.

Yet a further application scenario is introduced in more detail in the following since it is used to illustrate the results presented in this thesis.

[1] `http://www.franz.com/agraph/amdocs/`

Example 1.1. *Consider an online store selling audio CDs. In order to attract more clients and raise sales, the store manager decided to introduce more sophisticated tools for recommending and searching CDs.*

For that purpose, an ontology is used for structuring and maintaining the database of CDs. Each CD is associated with a unique identifier, a publisher, a release date, and the pieces of music the CD contains. Each piece of music has at least one track, and it is possible that a piece has several tracks (as is common for classical music). Additionally, each piece has a unique identifier and can be associated with the artist, composer, genre, origin of the piece.

Moreover, the system should be able to express guidelines for recommendations, either based on general criteria or based on customer specifications. For example, the store may want to automatically recommend to all customers CDs that are on offer or top sellers. Or some customer may want to get recommendations for CDs that he does not already own and that, according to some preference criteria, he probably likes. Whereas the first guideline can be represented in the ontology, the second one requires the closed world assumption (e.g., for inferring "by default", i.e., in the absence of evidence to the contrary, that the customer does not have the CD) and can be represented by a non-monotonic rule.

All of these examples demonstrate why application developers frequently voice that it would be favorable to have *local* closed world modeling as an additional feature for ontology-based systems. More precisely, it would be desirable to have a KRR formalism that allows us to interpret some parts of the knowledge base under the CWA, and others under the OWA. Such capabilities would also considerably enhance the usability of OWL.

1.4 The Problem: Combining Rules and Ontologies efficiently

Ontologies are a standard OWA formalism while non-monotonic rules apply the CWA. A combination of ontologies and rules[1] would clearly yield a combination of the OWA and the CWA. However, combining rules and ontologies is a non-trivial task, since even a naive combination of ontologies and OWA-based rules is already undecidable

[1]Note that ontologies and DLs are used rather synonymously in this context of a combination of OWA and CWA, while rules applying CWA always refer implicitly to non-monotonic rules.

[Horrocks and Patel-Schneider, 2004]. In fact, formalisms for rules and formalisms for ontologies differ substantially on how decidability is achieved. For ontologies, decidability is achieved by specific syntactic restrictions on the available first-order predicates, and by restricting the way these predicates can be related. Rule languages do not have such syntactic restrictions, but are usually limited in their applicability to the finitely many different objects explicitly appearing in the knowledge base. An immediate effect of these differences is that some expressive features of one of the approaches are not available in the other approach. Namely, rules make it possible to express: non-tree-shape-like relationships [Vardi, 1996][1], such as "an uncle is the brother of one's father"; integrity constraints [Reiter, 1992] to state, e.g., that a certain piece of information has to be explicitly present in the database; and closed world reasoning and specification of exceptions, as discussed in the previous section. Ontologies, on the contrary, make it possible to express open world reasoning, reason with unbounded or infinite domains, and they are thus well-suited to represent many types of incomplete information and schema knowledge. For example, in rule-based formalisms one typically cannot say that "every person has a father and a mother who are both persons" without listing all the parents explicitly. In Description Logics, this can be easily expressed by the following formula:

$$\texttt{Person} \sqsubseteq \exists \texttt{HasFather}.\texttt{Person} \sqcap \exists \texttt{HasMother}.\texttt{Person} \tag{1.4}$$

Our stance is that a combination of rules and ontologies is not only of interest for current applications in the Web, but also as a highly sophisticated means of knowledge representation in general. As argued in [Motik and Rosati, 2010], such a hybrid formalism combining rules and DL ontologies should satisfy certain criteria:

- Faithfulness: The integration of DLs and rules should preserve the semantics of both formalisms – that is, the semantics of a hybrid knowledge base in which one component is empty should be the same as the semantics of the other component. In other words, the addition of rules to a DL should not change the semantics of the DL and vice versa.

- Tightness: Rules should not be layered on top of a DL or vice versa; rather, the integration between a DL and rules should be tight in the sense that both the DL

[1] The DL $\mathcal{SROIQ}$ [Horrocks et al., 2006a] also provides role composition axioms, which can be used to address some, but by no means all, use cases.

and the rule component should be able to contribute to the consequences of the other component.

- Flexibility: The hybrid formalism should be flexible and allow one to view the same predicate under both open and closed world interpretation. This allows us to enrich a DL with non-monotonic consequences from rules, and to enrich the rules with the capabilities of ontology reasoning described by a DL.
- Decidability: To obtain a useful formalism that can be used in applications ,such as those in the Semantic Web, the hybrid formalism should be at least decidable, and preferably of low worst-case complexity.

Faithfulness is of clear advantage for augmenting already existing knowledge bases. E.g., given an ontology, its semantics remains the same in the new formalism unless we explicitly add rules. The new formalism is thus more compatible with each of its components and easier to grasp for knowledge base engineers trying to augment an ontology with rules or vice-versa.

In the case of tightness, it is sometimes argued that non-tight combinations are easier to handle on a technical level and that there are few applications requiring a tight integration. In fact, applications, such as the one on customer relation management in Section 1.3, seem to clearly distinguish the ontology as the basis and rules defined on top of the ontology that do only call for information from the ontology and never transfer knowledge to it. However, in cases where we want to model exceptions that have further impact on derivable knowledge, it is clearly preferable to have a tight combination of rules and ontologies. As an example, consider the example of people whose heart is on the right hand side. Clearly, this has further effects on the anatomy of the body of that person so that this conclusion has to be available in general to the entire knowledge base.

Flexibility separates approaches based on whether atoms can be interpreted under both OWA and CWA (or not). This property does trivially not hold for any approach that does not allow for the usage of the CWA. We have provided arguments for the usefulness of the CWA in the previous section. Moreover, it is more convenient to have a flexible formalism for modeling knowledge. Otherwise we would have to somehow maintain a mapping that relates corresponding predicates and we would have to ensure that these corresponding predicates are appropriately synchronized.

Finally, decidability has proven to be a useful property and is considered standard for description logics. Clearly, under certain circumstances, it is preferable to have a system that always solves the given reasoning task rather than one that is eventually more efficient but occasionally does not answer a query like most of the basic PROLOG interpreters that are only semi-decidable. Given the amount of data on the Web, it is clearly preferable to have a system that is not only decidable but also computationally as efficient as possible.

Thus, we follow [Motik and Rosati, 2010] and support these four criteria. In the next section, we review previous proposals for combining ontologies and rules (more general: open and closed world reasoning) and point out shortcomings w.r.t. these criteria. For a detailed comparison with our proposal we refer to Chapter 7.

1.5 Existing Combinations of Rules and Ontologies

Several proposals exist for combining rules and ontologies (see, e.g., [Drabent et al., 2009; Eiter et al., 2008a; Hitzler and Parsia, 2009] for a brief survey). They can be split into two groups, namely those semantically based on first-order logics solely, such as description logics alone, and the hybrid approaches providing a semantics combining elements of first-order logics with non-monotonicity.

The most general approach in the first group is the Semantic Web Rule Language (SWRL) [Horrocks and Patel-Schneider, 2004], an unrestricted combination of OWL DL with function-free Horn rules, i.e., rules without negation. The approach is very expressive but undecidable, yet nevertheless generalizes many approaches in this group. Applying, e.g., DL-safety to SWRL rules yields DL-safe rules [Motik et al., 2005], a decidable subset of SWRL. DL-safety ensures that rules are only applied to individuals that are explicitly present in the knowledge base. $\mathcal{AL}$-log [Donini et al., 1998], a combination of DL-safe positive rules and $\mathcal{ALC}$, and CARIN [Levy and Rousset, 1998] are also notable formalisms generalized by SWRL. In both cases, the ontology only serves as input to the rules and not vice versa, i.e., information flow is one way. Description Logic Programs (DLP) [Grosof et al., 2003] are a fragment of OWL that can be transformed into logic programs of positive rules. In the same spirit, Horn-$\mathcal{SHIQ}$ [Hustadt et al., 2005] is a fragment of OWL that can be translated into Datalog, and (like DLP)

is of tractable data complexity.[1] Recently, DLP has been generalized to Description Logic Rules [Krötzsch, 2010; Krötzsch et al., 2008a], i.e., rules that may contain description logic expressions. This enriches the DL, on which the description logic rules are based, with sophisticated constructs normally only available to more expressive description logics, without increasing the complexity. Similarly, ELP [Krötzsch et al., 2011; Krötzsch, 2010; Krötzsch et al., 2008b] is a polynomial language covering important parts of OWL 2. ELP also allows some axioms that cannot be expressed in OWL 2. All of these approaches have the advantage of fitting semantically into the original (first-order) OWL semantics, which also means that existing reasoners for ontologies alone can be used for reasoning in the combined knowledge bases. On the other hand, none of these approaches can express non-monotonic negation, and as such none cover the motivating cases discussed in Section 1.3. Therefore, none can satisfy the criterion of flexibility.

In the second group, the approach in [Eiter et al., 2008b] combines ontologies and rules in a modular way, i.e., both parts and their semantics are kept separate. The two reasoning engines nevertheless interact bidirectionally (with some limitations on the transfer of information) via interfaces, and the dlvhex system [Eiter et al., 2006] provides an implementation that generalizes the approach by allowing multiple sources for external knowledge (with differing semantics). This work has been extended in various ways, e.g., probabilities, uncertainty, and priorities; for references see the related work section of [Eiter et al., 2008b]. A related well-founded semantics [Eiter et al., 2004, 2011] has also been proposed that maintains the same modular interaction including the limitations on the transfer of information. Since the interaction between rules and ontologies is limited, the tightness criterion is not fully satisfied for all these modular approaches, and they are inflexible since predicates are completely modular even though it is possible to some extent to link predicates from ontologies to rules.

One of the few other well-founded semantics approach is called hybrid programs [Drabent and Małuszyński, 2007, 2010], which permits a form of reasoning by cases. It is therefore closer to a first-order semantics in some aspects. But this approach only allows the transfer of information from the ontology to the rules. Thus, hybrid programs are even less tight than [Eiter et al., 2011].

[1] Further analyses of Horn description logics are provided in [Krötzsch et al., 2007].

There are several further approaches related to stable models of logic programs. Hybrid MKNF knowledge bases [Motik and Rosati, 2010] tightly combine rules and ontologies in the unifying framework of the logics of Minimal Knowledge and Negation as Failure (MKNF) [Lifschitz, 1991]. In [de Bruijn et al., 2007a], a variety of embeddings into auto-epistemic logic is used to tightly combine ontologies and rules. This approach is quite similar in spirit to hybrid MKNF [Motik and Rosati, 2010]. However, a precise relation to hybrid MKNF is far from obvious since an autoepistemic interpretation in [de Bruijn et al., 2007a, 2011] is a pair of a first-order interpretations and a set of beliefs, and both are not necessarily related. Moreover, $\mathcal{DL}$+log [Rosati, 2006] provides a combination of rules and ontologies that separates predicates into rule and ontology predicates and evaluates the former w.r.t. the answer set semantics and the latter w.r.t. a first-order semantics with weak DL-safety, i.e., each variable in the head of a rule appears in an arbitrary positive atom in the body of the rule. This separation of predicates means that the approach is not flexible. Like [Motik and Rosati, 2010], [de Bruijn et al., 2007b] generalizes [Rosati, 2006] and several earlier related works (e.g., [Rosati, 2005]) within the framework of equilibrium logics. Quite similar to [Rosati, 2006] is [Lukasiewicz, 2007, 2010], although this approach does not distinguish between ontology and rules predicates. In fact, the work originates from [Eiter et al., 2008b], but it permits a much tighter integration. However, it turns out to be not faithful w.r.t. the first-order semantics of DLs. A related well-founded semantics is also defined [Lukasiewicz, 2010]. Open answer set programming [Heymans et al., 2007] extends rules with open domains and adds some syntactic limitations for ensuring decidability. Based on that, an algorithm has been provided for f-hybrid knowledge bases [Feier and Heymans, 2009], i.e., a combination of ontologies and rules without DL-safety but which limits to predicates that satisfy a tree-shaped property. Thus, relations as the previously mentioned example of an uncle are not expressible in this approach. A loose layering of PROLOG on top of DLs, employing four-valued logic, is presented in [Matzner and Hitzler, 2007]. This approach is of course not tight.

An alternative way of combining open and closed world reasoning is to enrich DLs with further syntactic constructs representing non-monotonic features. Among these approaches, Description Logics of MKNF [Donini et al., 2002], which allows two modal operators in ontology axioms, is quite closely related to [Motik and Rosati, 2010]. An algorithm is provided in [Donini et al., 2002] for $\mathcal{ALC}$ with MKNF, and it has

been improved in [Ke and Sattler, 2008]. In [Bonatti et al., 2006, 2009; Grimm and Hitzler, 2009; Krisnadhi et al., 2011b], circumscription is used for adding non-monotonic reasoning to DLs, and several other formalisms introducing defaults to ontologies exist (e.g., [Baader and Hollunder, 1995]). Since all these approaches are not based on rules, we do not consider them any further. We justify this decision to focus on rules with the fact that rules are more commonly used than the previously mentioned non-monotonic formalisms, and we claim that this increases the support and acceptance of an approach combining open and closed world reasoning. Additionally, relying on rules for closed world reasoning might simplify the implementation of the combination. Whenever reasoning in rules and ontologies is to some extent modular, and for several of the presented approaches for combining rules and ontologies this is the case, even for the tightly integrated ones, then previous work on implementations might be (at least partially) reusable while the approaches enriching DLs require to alter the details of its implementations.

1.6 A Novel Approach

As shown in [Motik and Rosati, 2010], among the various proposals for combining rules and ontologies, the only one satisfying all four criteria presented in Section 1.4 are Hybrid MKNF knowledge bases. Hybrid MKNF knowledge bases [Motik and Rosati, 2010] seamlessly integrate arbitrary decidable description logics with (disjunctive) logic programming rules, making it possible to reason over a combination of monotonic open world knowledge and non-monotonic closed world knowledge within a single (hybrid) framework. A detailed discussion about the importance of Hybrid MKNF knowledge bases for modeling knowledge in the Semantic Web can be found in [Horrocks et al., 2006b], and [Grimm and Hitzler, 2008; Grimm et al., 2006] provide arguments for the usefulness of epistemic reasoning in the way it is done in MKNF logics.

Several reasoning algorithms are presented in [Motik and Rosati, 2010] for Hybrid MKNF knowledge bases, and it is shown that the data complexity of reasoning within this framework is in many cases not higher than reasoning in the corresponding fragment of logic programming. Thus, adding an ontology to rules does not in general increase the data complexity when compared to rules alone. But the same cannot be said about adding rules to ontologies. E.g., we have at least a data complexity of coNP

for a combination of normal logic programming rules with ontologies even if the data complexity of the Description Logics fragment is in the complexity class P. Indeed, although the approach of Hybrid MKNF knowledge bases is powerful, whenever we add rules with arbitrary non-monotonic negation to an ontology, we in general loose tractability. Only a specific limited use of non-monotonic negation, namely in the form of stratified rules, admits to maintain tractability (see [Motik and Rosati, 2010]). However, we claim that robustness w.r.t. updates and the combination of different sources of information is an important property of a combination of rules and ontologies. Since it cannot be guaranteed that this property is maintained in such cases, we obtain a higher computational complexity in general.

The reason for that increase in the complexity lies in the fact that, as shown in [Lifschitz, 1991], rules are interpreted in a similar way as in the Stable Model Semantics [Gelfond and Lifschitz, 1988] for logic programs, whose reasoning algorithms are NP-hard. So, if a semantics based on the SMS is adopted, then any improvements on the complexity of the combination of rules and ontologies are bound by NP-hardness [Dantsin et al., 2001].

The other major semantics for normal logic programs – the Well-Founded Semantics [van Gelder et al., 1991] – seems to offer a solution. WFS is a three-valued semantics, where propositions can be 'true', 'false' or 'undefined' (while in SMS propositions can only be 'true' or 'false'), and WFS assigns a single model – the well-founded model – to every non-disjunctive logic program. The WFS is sound with respect to the SMS, in that whenever a proposition is true (resp. false) under the WFS, then it is also true (resp. false) in all stable models. Though the WFS is semantically weaker than SMS (in terms of the derivable true and false consequences), reasoning in the WFS has a lower computational complexity than in SMS – for normal programs the data complexity is P for the WFS instead of coNP for SMS [Dantsin et al., 2001]. Our stance is that the lower complexity bound makes WFS more promising than SMS as a basis for the semantics of hybrid knowledge bases. This is even more the case in application areas, such as the ones mentioned above in Section 1.3, where huge amounts of data are involved.

Additionally, reasoning in SMS requires one to obtain the entire model of a knowledge base (just like [Motik and Rosati, 2010] for combinations of rules and ontologies), while the WFS is amenable to top-down, query-driven reasoning, in which only the

part of the knowledge base "relevant" to a specific query is accessed [Chen and Warren, 1996]. This makes a WFS based approach all the more suitable for large scale applications.

1.7 Contributions

In this thesis, we define a new semantics for Hybrid MKNF knowledge bases, restricted to non-disjunctive rules, that soundly approximates the semantics of [Motik and Rosati, 2010] but is, in some important cases, in a strictly lower complexity class. In particular, when dealing with a tractable description logic, our combined approach remains tractable w.r.t. data complexity. We achieve this by extending the two-valued MKNF semantics from [Motik and Rosati, 2010] to three truth values, where each two-valued model from [Motik and Rosati, 2010] corresponds to a total three-valued model of our approach (and vice versa), and where the least (w.r.t. derivable knowledge) three-valued MKNF model is the well-founded MKNF model. Our proposal straightforwardly satisfies the four criteria presented above for the combination of rules and ontologies. Moreover, the proposed semantics also guarantees the following properties:

- The well-founded MKNF model is sound w.r.t. the two-valued MKNF models of [Motik and Rosati, 2010], i.e., each query that is true (respectively false) in the well-founded MKNF model is also true (respectively false), in each two-valued MKNF model.

- Our proposal coincides with the original DL-semantics when no rules are present, and the original WFS of logic programs if the DL component is empty.

- If the knowledge base is consistent, then the approach is coherent in the sense of [Pereira and Alferes, 1992], i.e., if a formula φ is first-order false in the ontology, then the non-monotonic interpretation of φ in the rules is enforced to be false as well.

- If the knowledge base is inconsistent, then our approach allows us to detect inconsistencies without any substantial additional computational effort.

- The computational data complexity of our approach depends on the computational complexity of the applied DL, but if the considered DL is of polynomial data complexity, then the combination with rules remains polynomial.

We present an algorithm based on an adaptation of the alternating fixpoint for logic programs whose calculated result is a representation of the well-founded MKNF model. Additionally, we devise an alternative characterization based on unfounded sets (as known from logic programs) and show that the calculated result of that characterization coincides with the one obtained from the alternating fixpoint construction.

We also present a querying mechanism, called **SLG**($\mathcal{O}$), that is sound and complete for our proposed semantics and sound for the two-valued semantics of Hybrid MKNF knowledge bases [Motik and Rosati, 2010]. **SLG**($\mathcal{O}$) accepts DL-safe conjunctive queries, i.e., conjunctions of predicates with variables where queries have to be ground when processed in the ontology, returning all correct answer substitutions for variables in the query. Queries itself are basically evaluated in a top-down manner based on SLG resolution. This means that if a queried atom matches the conclusion of the rule, then we query for the premise. In this way, the search is continued until an answer has been found. Ontologies are incorporated into this process using so-called oracles. Whenever the queried atom appears in the ontology, then we may query the oracle of the ontology whose answer consists of a set of atoms that if proven ensure the truth of the originally queried atom.

The defined general procedure applies to any DL and under certain conditions maintains the data complexity of our new proposal. Then, we also provide concrete oracles for $\mathcal{REL}$, DLP and *DL-Lite*, i.e., three tractable fragments that are underlying the OWL 2 profiles that are part of the W3C recommendations [Hitzler et al., 2009a] for the Semantic Web. We show that the oracles thus defined are correct with respect to the general procedure and maintain the polynomial data complexity.

1.8 Outline

The thesis is divided into three parts. The first part comprises of four chapters starting with the current chapter that introduces and motivates our work. In Chapter 2, after recalling some general notions including a section on computational complexity, we present the common syntax of logic programs and several different semantics that are

important for our work. In Chapter 3 we present the syntax and the semantics of the expressive description logic $\mathcal{SROIQ}$. We also present the reasoning tasks and three tractable fragments of $\mathcal{SROIQ}$. Finally, in Chapter 4, we present the logics of minimal knowledge and negation as failure and the formalism of hybrid MKNF knowledge bases. This finishes the part providing an overview of the field in which we present the results of our work. Note that readers familiar with Description Logics or Logic Programming may skip the corresponding chapter, but we want to point out that the notions presented in the first two sections of Chapter 2 are not limited to this chapter or LP.

In the second part, we present the new well-founded MKNF semantics. In Chapter 5, we introduce a three-valued MKNF semantics. We extend the evaluation of MKNF formulas and the notion of MKNF model to three truth values. Then, we determine a unique minimal model among all three-valued MKNF models, the well-founded MKNF model. In Chapter 6, we provide an algorithm that, as it turns out, allows us to compute a representation of that unique well-founded MKNF model. This alternating fixpoint computation also admits the detection of inconsistencies in the combined knowledge base, and we show that several desired properties, such as faithfulness, are satisfied. In Chapter 7, we compare in more detail our new approach with related work.

The contributions presented in the second part were published in the International Workshop on Description Logics [Knorr et al., 2007a], in the Portuguese Conference on Artificial Intelligence [Knorr et al., 2007b], in the European Conference on Artificial Intelligence [Knorr et al., 2008], and in Artificial Intelligence [Knorr et al., 2011].

In the third part, we present a top-down querying procedure for the newly defined semantics. In Chapter 8, we present the general framework $\mathbf{SLG}(\mathcal{O})$, which extends SLG resolution with the capability to query ontologies. It is shown that the answers to queries posed to a consistent MKNF knowledge base correspond to the unique well-founded MKNF model. Then, in Chapter 9, we also consider specific tractable combinations for that querying mechanism, namely combinations of rules with $\mathcal{REL}$, $DL\text{-}Lite_{\mathcal{R}}$, and DLP.

The work in this part has been published in the International Semantic Web Conference [Alferes et al., 2009] and in the European Conference on Artificial Intelligence [Knorr and Alferes, 2010].

The subsequent index contains pointers to definitions of notions and notations used in the thesis to ease the reading.

2

Rules in Logic Programming

Sets of rules in Logic Programming are a well-known means for non-monotonic knowledge representation and reasoning. In fact, over the last thirty five years, LP has been intensively studied and a large body of theoretical results has been presented accompanied by a variety of quite different systems aiming to implement the declarative paradigm underlying LP. In LP, non-monotonic reasoning is achieved by the usage of default negation (or negation as failure) in rules. The simplicity of this approach and the fact that LP is widely recognized as one of the most important non-monotonic reasoning formalisms are the reasons why we consider LP for the CWA in our combination of open and closed world reasoning. In this chapter, to make the presentation self-contained, we recall the notation and important definitions of LP with a focus on the semantics that are important for our work. A full account of semantics of logic programs is beyond the scope of this thesis, and we refer to the extensive list of references in, e.g., [Lobo et al., 1992; Minker and Seipel, 2002].

Here, we recall Answer Set Programming [Gelfond and Lifschitz, 1991], which is relevant for our work since this general approach is the one to which a correspondence result has been established with hybrid MKNF knowledge bases [Motik and Rosati, 2010] – the approach that forms the starting point of our work for combining non-monotonic rules and ontologies. Building on the minimal semantics [Minker, 1982], ASP extends the two-valued Stable Model Semantics [Gelfond and Lifschitz, 1988] with disjunctions [Przymusinski, 1991], and it allows two kinds of negation (classical and default) in rules. The SMS itself is only defined for a less expressive class of logic programs, but the established relation [van Gelder, 1989] to the other standard

semantics for normal logic programs, namely the three-valued Well-Founded Semantics [van Gelder et al., 1991], is of importance.

In our work, which is based on the Well-founded Semantics, we focus exactly on the class of normal programs, which does not allow for disjunction or classical negation. There are also well-founded semantics defined for logic programs allowing classical negation [Pereira and Alferes, 1992] and logic programs allowing disjunctions in the conclusion of the rules[Alcântara et al., 2005; Baral et al., 1990; Brass and Dix, 1998; Ross, 1990; Wang, 2001]. However, as we argue in this thesis, there are strong reasons not to consider disjunction in combination with well-founded semantics in general, and even more in the context of the hybrid approach based on MKNF. In case of classical negation, we show why a more complex semantics including classical negation in rules is not needed in the context of our work on hybrid MKNF. Intuitively, reasoning on classical negation can be transfered to the ontology. Both these issues are discussed in Chapter 4.

We start recalling some notions related to fixed point semantics that are used in this chapter but also in the rest of the thesis (Section 2.1) and some notions on computational complexity (Section 2.2). Then, we present the syntax and some terminology of logic programs as used in ASP (Section 2.3). We continue with the semantics of ASP (Section 2.4) and then we recall SMS, as one concrete subset[1] (Section 2.5). We finish this chapter by recalling the well-founded semantics and the relation to SMS (Section 2.6).

2.1 Fixed Point Semantics for Logic Programs

Logic programming semantics are quite often based on iterations of some operators, and the semantics we recall in this chapter are no exception to that. To make the presentation of these semantics self-contained, we now recall some notions and theoretical results related to fixed points. We would like to point out that we limit ourselves to the notions that are going to be used explicitly in this thesis, and we refer to [Hitzler and Seda, 2010] for a comprehensive book on fixed point approaches in logic programming.

[1] We would like to point out that presenting ASP before SMS is rather uncommon. In this aspect, we follow the idea of presenting first the general and more expressive form, and then restrict to a concrete subset which usually provides us with some benefit for efficiency of computation.

The basic idea is that operators, i.e., functions, immediately derive information based on the logic program and the knowledge assumed to be true. One commonly applied scheme is to start with applying the operator to the empty set, and repeatedly use the outcome of such a derivation as input for another iteration of the operator. An important property of such iterations in LP is monotonicity and we recall it and antitonicity as follows.

Definition 2.1. *Let X be the power set of a given set S. A function f is called* monotonic *if, for all $x, y \in X$, it holds that $x \leq y$ implies $f(x) \leq f(y)$, and f is called* antitonic *if , for all $x, y \in X$, it holds that $x \leq y$ implies $f(y) \leq f(x)$.*

Thus, if an iteration is monotonic, then the sequence of subsequent results is strictly increasing. An antitonic operator is commonly used in LP to define another operator that results from applying the original operator itself twice, and it can be shown that such a doubled operator is again monotonic. Monotonicity of a function is crucial in the simplified version of the well-known result of fixpoints [Tarski, 1955] that shows that the sequence eventually reaches a point at which it stops increasing.

Theorem 2.2. *Let X be the power set of a given set S, $f : X \to X$ monotonic, and $x, y \subseteq S$. Then f has a* least fixpoint $x = f(x)$ *and a* greatest fixpoint $y = f(y)$.

This result has been repeatedly applied in logic programming semantics in the sense that a monotonic operator is defined that can be iterated until a fixpoint is reached that corresponds to the information derivable from the program. Note that such fixpoints are not restricted to finite iterations in LP, which means that natural numbers are not sufficient to represent the number of iterations. Instead ordinals, an extension of natural numbers, are used. For completeness, we recall that notion, but we refer to [Hitzler and Seda, 2010] for the details.

Definition 2.3. *An* ordinal *is an equivalence class of a well-ordering under isomorphism. A* successor ordinal *is an ordinal α such that there is a greatest ordinal β with $\beta < \alpha$. In this case α is the* successor *of β and can be denoted $\beta + 1$. Any other ordinal is called* limit ordinal.

Intuitively, we may consider ordinals as a way of capturing uncountable sequences. Natural numbers are ordinals where each $n \in \mathbb{N}$ with $n > 0$ represents a successor ordinal whereas 0 is a limit ordinal. The limit ordinal for natural numbers is $\omega = \{\beta \mid$

$\beta < \omega$ and $\beta \in \mathbb{N}\}$. Then, $\omega + 1$, $\omega + 2$, ... are successor ordinals of ω, and $\omega * 2$ is the next limit ordinal. There are uncountably many ordinals, but, for our focus on iterations in LP semantics, the material presented above suffices.

For proving properties of ordinals we recall the principle of transfinite induction that extends induction on natural numbers to ordinals.

Definition 2.4. *The* principle of transfinite induction *is given in the following: Suppose we want to prove that a property Q holds for all members of an ordinal α. Then it suffices to show that the following hold.*

(i) $Q(0)$ holds.

(ii) For any ordinal β, if $Q(\alpha)$ holds for all ordinals $\alpha < \beta$ then $Q(\beta)$ holds as well.

Often, part (ii) is divided into successor ordinals and limit ordinals.

2.2 Computational Complexity

In this section we briefly recall the relevant notions of computational complexity that are used in this thesis. We would like to point out that a full introduction to the topic would exceed the scope of this thesis. Instead, we recall only the notions that are relevant for our work, and we refer to [Papadimitriou, 1994] for an extensive introduction to computational complexity.

Intuitively, we may understand computational complexity as measuring the computational effort to solve some problem and relating it to a complexity class of corresponding problems. Complexity is usually measured as worst case complexity, even though, in practice, systems might achieve better results on average. A problem refers to certain reasoning tasks that either reply only with yes or no (decision problems) or require a more complex answer (function problems). In both cases, the computational complexity is measured as a function depending on the input. The input is the size of the logical theory in consideration (DL knowledge base, a set of rules, or its combination), and the size itself refers to the minimal number of symbols necessary to write the theory in the alphabet provided by its signature, including logical operators and auxiliary symbols, such as parentheses.

It is common to divide a theory into axioms describing the domain and assertions that describe a concrete problem. In computational complexity, we may measure only

the data complexity w.r.t. the assertions or the taxonomic complexity w.r.t. domain axioms. The combined complexity measures the combination of the two, sometimes also taking into consideration the size of the queries if these occur in the problem in consideration.

In our work we encounter the complexity classes LogSpace, PTime, NP, coNP, DP, NExpTime, and N2ExpTime and we refer to [Papadimitriou, 1994] for their standard definitions. Here, we simply recall the brief definitions that are mainly taken from [Motik et al., 2009a].

- LogSpace: the class of problems solvable by a deterministic algorithm using space that is at most logarithmic in the size of the input, i.e., roughly $\log(n)$, for n the size of the input.
- PTime: the class of problems solvable by a deterministic algorithm using time that is at most polynomial in the size of the input, i.e., roughly n^c, for n the size of the input and c a constant.
- NP: the class of problems solvable by a nondeterministic algorithm using time that is at most polynomial in the size of the input, i.e., roughly n^c, for n the size of the input and c a constant.
- coNP: the class of problems whose complements are in NP.
- DP: the class of problems that contain one problem in NP and one in coNP.
- NExpTime: the class of problems solvable by a nondeterministic algorithm in time that is at most exponential in the size of the input, i.e., roughly 2^n, for n the size of the input.
- N2ExpTime: the class of problems solvable by a nondeterministic algorithm in time that is at most double exponential in the size of the input, i.e., roughly 2^{2^n}, for n the size of the input.

Most of these complexity classes subsume each other in the order of appearance, N2ExpTime being the most complex one, with some notable exceptions such as NP and coNP. However, it has not been proven in all cases whether the inclusion is strict or an identity holds. A famous example for this problem are PTime and NP. The

complexity class PTime (and also each lower class including LogSpace) is commonly called *tractable*, while all classes above PTime are considered intractable.

Reductions are used to show that a problem is contained in a certain complexity class. In this sense, a problem P is called *hard* for a class $\mathcal{C}$ if every problem in $\mathcal{C}$ can be reduced to P. A problem is called *complete* for some complexity class $\mathcal{C}$ if tight complexity bounds are known – that is, the problem is known to be both in the complexity class $\mathcal{C}$, i.e., an algorithm is known that only uses time/space in $\mathcal{C}$, and hard for $\mathcal{C}$. For a class $\mathcal{C}$, we may say that a problem is $\mathcal{C}$-hard or $\mathcal{C}$-complete.

One generalization of complexity classes for the problems that have some polynomial algorithm is achieved by the polynomial hierarchy. For complexity classes $\mathcal{C}$ and $\mathcal{E}$, with $\mathcal{E}^{\mathcal{C}}$ we denote the class of decision problems that can be solved by a Turing machine running in $\mathcal{E}$, using an oracle for decision problems in $\mathcal{C}$. The polynomial hierarchy is defined inductively as follows:

$$\Delta_0^p = \Sigma_0^p = \Pi_0^p = \text{PTime}, \quad \Delta_{k+1}^p = \text{PTime}^{\Sigma_k^p}, \quad \Sigma_{k+1}^p = \text{NP}^{\Sigma_k^p}, \quad \Pi_{k+1}^p = \text{coNP}^{\Sigma_k^p}.$$

In particular, the classes for $k = 0$ correspond to PTime, NP, and coNP.

2.3 Terminology of Logic Programs

Logic programs are based on a signature containing the basic elements to form the expressions that appear in logic program rules. Such a signature consists of three disjoint sets, namely the sets of constants, function symbols, and predicate names.

Definition 2.5. *A* signature $\Sigma = (\Sigma_c, \Sigma_f, \Sigma_p)$ *consists of three disjoint sets of constants* Σ_c*, function symbols* Σ_f*, and predicates* Σ_p*.*

Given such a signature and a set of variables, we are able to define atoms and the terms appearing therein.

Definition 2.6. *Let* $\Sigma = (\Sigma_c, \Sigma_f, \Sigma_p)$ *be a signature and* Σ_v *a disjoint set of variables. A* term *is a constant* $c \in \Sigma_c$*, a variable* $v \in \Sigma_v$*, or a compound term* $f(t_1, \ldots, t_n)$ *where* f *is a function symbol of arity* n *and all* t_i*,* $1 \leq i \leq n$ *are terms. If* $n = 0$*, then the compound term is equivalent to a constant* $c \in \Sigma_c$*. A* (first-order) atom *is of the form* $p(t_1, \ldots, t_n)$*, where* p *is an* n*-nary predicate symbol and the* t_i*, for* $i = 1, \ldots, n$ *and* $n \geq 0$*, are terms.*

Such atoms allow us to express that $\mathtt{x}$ is the uncle of $\mathtt{y}$ using $\mathtt{uncle}(\mathtt{x},\mathtt{y})$. As another example, we may consider $\mathtt{smallerthan}(\mathtt{0},\mathtt{s}(\mathtt{0}))$ to express that 0 is smaller than its successor. If we want to express the opposite of such information, then we may either use classical negation, e.g., to express that $\mathtt{x}$ is not the uncle of $\mathtt{y}$ using $\neg\mathtt{uncle}(\mathtt{x},\mathtt{y})$ or default negation to express that $\mathtt{x}$ is not the uncle of $\mathtt{y}$ unless further information allows us to derive the opposite using $\mathbf{not}\,\mathtt{uncle}(\mathtt{x},\mathtt{y})$. We may even express that, by default, $\mathtt{x}$ is not the uncle of $\mathtt{y}$ unless further information allows us to derive the opposite using $\mathbf{not}\,\neg\mathtt{uncle}(\mathtt{x},\mathtt{y})$.

Some notions related to the usage of negation in logic programs is presented in the next definition.

Definition 2.7. *Let $\Sigma = (\Sigma_c, \Sigma_f, \Sigma_p)$ be a signature and Σ_v a disjoint set of variables. A literal, is either a* positive literal*, i.e., an atom $p(t_1,\ldots,t_n)$ or a classically negated atom $\neg p(t_1,\ldots,t_n)$, or a* negative literal*, i.e., a positive literal preceded by* **not** *.*

Note that $\neg p(t_1,\ldots,t_n)$ is not always considered positive in the literature, but we follow in that aspect the terminology of ASP.

Logic programs consist of rules that combine (possibly negated) atoms in logic formulas of a specific form.

Definition 2.8. *Let $\Sigma = (\Sigma_c, \Sigma_f, \Sigma_p)$ be a signature and Σ_v a disjoint set of variables. A* generalized disjunctive (logic) program Π *consists of finitely many universally quantified* generalized disjunctive rules *of the form*

$$H_1 \vee \cdots \vee H_l \leftarrow A_1 \wedge \cdots \wedge A_n \wedge \mathbf{not}\, B_1 \wedge \cdots \wedge \mathbf{not}\, B_m \tag{2.1}$$

usually written as

$$H_1 \vee \cdots \vee H_l \leftarrow A_1, \ldots, A_n, \mathbf{not}\, B_1, \ldots, \mathbf{not}\, B_m \tag{2.2}$$

where H_k, A_i, and B_j, for $k = 1,\ldots,l$, $i = 1,\ldots,n$, and $j = 1,\ldots,m$, are positive literals defined over Σ and Σ_v, and where the symbol **not** *represents* default negation*. A rule can be divided into the* head $H_1 \vee \cdots \vee H_l$ *and the* body $A_1,\ldots,A_n$, $\mathbf{not}\, B_1,\cdots,\mathbf{not}\, B_m$*. If the body is empty, i.e., $n = m = 0$, the rule is called a* fact*.*

We also identify the elements in the head and in the body with the sets $\mathcal{H} = \{H_1,\ldots,H_l\}$, $\mathcal{B}^+ = \{A_1,\ldots,A_n\}$, and $\mathcal{B}^- = \{\mathbf{not}\, B_1,\ldots,\mathbf{not}\, B_m\}$, where $\mathcal{B} = \mathcal{B}^+ \cup \mathcal{B}^-$, and we occasionally abbreviate rules with $\mathcal{H} \leftarrow \mathcal{B}^+ \wedge \mathcal{B}^-$ or even $\mathcal{H} \leftarrow \mathcal{B}$.

If, for all rules in program Π, $l = 1$, $m = 0$, *and only atoms are allowed in* Π, *then* Π *is a* definite (logic) program *consisting of* definite rules. *Given a set of definite rules that additionally allow the appearance of default negation in the body, i.e.,* $m \geq 0$, *yields* normal rules *and* normal (logic) programs. *Normal programs with disjunctions, i.e.,* $l \geq 1$, *are* disjunctive (logic) programs *consisting of* disjunctive rules, *and normal programs admitting also classical negation are called* generalized (logic) programs *consisting of* generalized rules. *A* positive program *can be any of the previous programs where each rule is* positive, *i.e., does not contain default negation. A rule is called* safe *if each variable in the rule occurs in an atom* A_i *for some* i, $1 \leq i \leq n$.

Thus, different kinds of logic programs yield a different expressiveness depending on how many and which literals are allowed in the head and the body of each rule.

We assume in the following that all rules are safe. Note that the disjunction in the head and the conjunction in the body are commutative, i.e., the order of the elements in the head and in the body does not matter.

The variables appearing in rules can be substituted by other terms including the special case where the result is free of variables. In this case, any expression, i.e., term, atom, literal, rule, or program, is called *ground* if it contains no variables. With that in mind, we can define some standard LP notions.

Definition 2.9. *Let* $\Sigma = (\Sigma_c, \Sigma_f, \Sigma_p)$ *be a signature for program* Π. *The* Herbrand universe HU_Π *is the set of all ground terms that can be formed from the constants from* Σ_c *and the function symbols from* Σ_f *occurring in* Π. *If there is no constant in* Π *then an arbitrary constant is added. The* Herbrand base HB_Π *is the set of all ground atoms that can be formed by using the predicate symbols from* Π *with terms as arguments from* HU_Π. *Finally, the set of all ground instances of the rules of a program* Π *with respect to* HB_Π *is denoted by* $\mathsf{ground}(\Pi)$.

The intuitive idea behind the Herbrand base is that it contains all ground atomic information that is possibly derivable from a given program. As such, it serves as the basis for the set on which operators are defined as introduced in Section 2.1.

Note that the notions in Definition 2.9 may not be finite because of the possible occurrence of function symbols with arity greater than 0, even though the program itself is only a finite set of (non-ground) rules. In practice, it is usual to restrict first-order atoms to be free of function symbols to achieve decidability, and, unless otherwise stated, we assume in the rest of this chapter that $\Sigma_f = \emptyset$.

2.4 Answer Set Semantics

The semantics of ASP can be obtained from the standard first-order semantics considering only Herbrand Models [Fitting, 1996]. Herbrand models are essentially interpretations that do interpret all ground terms by themselves. This obviously relates to the notion of Herbrand base and we can define the notion of interpretation in the context of ASP.

Definition 2.10. *Let* Π *be a generalized disjunctive logic program. An* interpretation *I of Π is a subset of $HB_\Pi \cup \{\neg A \mid A \in HB_\Pi\}$ such that there is no ground atom A with $A \in I$ and $\neg A \in I$.*

Interpretations are just subsets of the atoms and classically negated atoms appearing in the program in consideration. It is common to represent these subsets by mappings from the Herbrand base HB_Π to the set of truth values $\{\mathbf{t}, \mathbf{f}\}$, representing the two truth values true and false, hence the name two-valued logics. Those elements appearing in the interpretation are considered true, all the others false.

Such interpretations can be used to define answer sets straightforwardly for positive programs, minimizing the knowledge that is interpreted to true in order to satisfy the rules of the program.

Definition 2.11. *Let* Π *be a positive generalized disjunctive logic program. An interpretation I* satisfies *a positive ground rule r in* $\mathsf{ground}(\Pi)$ *of the form $H_1 \vee \cdots \vee H_l \leftarrow A_1, \ldots, A_n$, written $I \models r$, if $A_i \in I$, for all $1 \leq i \leq n$, implies, for some $1 \leq k \leq l$, $H_k \in I$. I* satisfies *Π, written $I \models \Pi$ if and only if $I \models r$ for each r in* $\mathsf{ground}(\Pi)$*. We call I an* answer set *for Π if $I \models \Pi$ and there is no $I' \subsetneq I$ such that $I' \models \Pi$.*

Thus, an interpretation I is an answer set for a positive generalized disjunctive logic program Π if all rules of Π are satisfied in I and I is minimal, i.e., there is no subset of I that also satisfies all the rules in Π.

This notion of an answer set can be generalized to programs that are not restricted to positive rules.

Definition 2.12. *Let* Π *be a generalized disjunctive logic program. The* transform *of Π w.r.t. an interpretation I is the positive ground program Π/I that is obtained from* $\mathsf{ground}(\Pi)$ *by deleting each rule of the form 2.2 from* $\mathsf{ground}(\Pi)$ *such that, for some $1 \leq j \leq m$, $B_j \in I$, and by deleting all remaining* **not** *B_j in the other rules. I is an* answer set *of Π if it is an answer set of Π/I.*

This definition is an extension of the so-called Gelfond-Lifschitz transform [Gelfond and Lifschitz, 1988]. An interpretation I is guessed and the negative literals are processed w.r.t. I, i.e., contradictory information leads to the removal of the entire rule, while all negative literals that are true in I are simply removed. The resulting program is positive and if I satisfies all the rules in the reduced program Π/I and is minimal, then I is an answer set.

Example 2.13. *We use the following program* Π *to exemplify answer sets.*[1]

$$\mathtt{a} \leftarrow \mathbf{not}\ \mathtt{b} \tag{2.3}$$

$$\mathtt{b} \leftarrow \mathbf{not}\ \mathtt{a} \tag{2.4}$$

$$\mathtt{c} \vee \neg\mathtt{c} \leftarrow \mathbf{not}\ \mathtt{a} \tag{2.5}$$

$$\mathtt{d} \leftarrow \mathtt{b}, \mathbf{not}\ \neg\mathtt{c} \tag{2.6}$$

The Herbrand base HB_Π *is* $\{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}$*. We obtain three answer sets, namely* $I_1 = \{\mathtt{a}\}$*,* $I_2 = \{\mathtt{b}, \mathtt{c}, \mathtt{d}\}$*, and* $I_3 = \{\mathtt{b}, \neg\mathtt{c}\}$*. E.g.,* $I_4 = \{\mathtt{a}, \mathtt{d}\}$ *also models* Π/I_4 *but it is not minimal:* $\mathtt{d}$ *is not true since* $\mathtt{b}$ *is not true. The same holds for* $I_5 = \{\mathtt{b}, \neg\mathtt{c}, \mathtt{d}\}$*:* $\neg\mathtt{c}$ *is true and the rule 2.6 is removed from the transform* Π/I_5*.*

2.5 Stable Model Semantics

The stable model semantics [Gelfond and Lifschitz, 1988] can be straightforwardly derived from the just presented answer set semantics by appropriately restricting the definitions in the previous section to normal logic programs. Nevertheless, we recall the semantics here because the simplifications that can be achieved with the restriction to normal logic programs are useful for establishing a relation to the three-valued well-founded semantics for normal logic programs in Section 2.6.

In analogy to the previous section, we could define how to obtain an answer set for normal programs that are positive. However, these programs coincide with definite programs and it turns out that there are not several minimal models but only one least model that can be computed by means of an operator. We recall that immediate consequence operator from [van Emden and Kowalski, 1976]. Note that interpretations

[1] For simplicity, quite a few of the technical examples are given in propositional logic. However, we can always embed these examples into first-order logic (or a fragment of it) by considering nullary predicates.

I are now just subsets of HB_Π since classical negation does not occur in normal logic programs.

Definition 2.14. *Let* Π *be a definite program. The operator* T_Π*, is defined by setting* $T_\Pi(I) = \{H \mid H \leftarrow A_1, \ldots, A_n \in \mathsf{ground}(\Pi)$ *and* $A_i \in I$ *for all* $1 \leq i \leq n\}$.

For definite logic programs, T_Π is ω-continuous [Lloyd, 1987] and allows us to obtain the least model of a definite logic program by re-iterating the operator. We define $T_\Pi \uparrow 0 = \emptyset$, $T_\Pi \uparrow (n+1) = T_\Pi(T_\Pi \uparrow n)$ and $T_\Pi \uparrow \omega = \bigcup_{i \geq 0} T_\Pi \uparrow i$, and the least model is obtained as the least fixed point of this operator.

Theorem 2.15. *Given a definite program* Π*, the* least model *is obtained as the least fixed point of* T_Π.

The proof can be found for example in [Lloyd, 1987].

Thus, we can actually compute the least model, while answer sets for positive generalized disjunctive logic programs have to be guessed and checked.

Example 2.16. *Let* Π *be the given program.*

$$\mathtt{a} \leftarrow \tag{2.7}$$

$$\mathtt{b} \leftarrow \mathtt{a} \tag{2.8}$$

$$\mathtt{c} \leftarrow \tag{2.9}$$

$$\mathtt{d} \leftarrow \mathtt{b}, \mathtt{c} \tag{2.10}$$

$$\mathtt{e} \leftarrow \mathtt{e} \tag{2.11}$$

We compute the least model using the operator T_Π.

$$\begin{aligned}
T_\Pi \uparrow 0 &= \emptyset.\\
T_\Pi \uparrow 1 &= T_\Pi(\emptyset) = \{\mathtt{a}, \mathtt{c}\}.\\
T_\Pi \uparrow 2 &= T_\Pi(\{\mathtt{a}, \mathtt{c}\}) = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}.\\
T_\Pi \uparrow 3 &= T_\Pi(\{\mathtt{a}, \mathtt{b}, \mathtt{c}\}) = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}.\\
T_\Pi \uparrow 4 &= T_\Pi(\{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}) = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\} = T_\Pi \uparrow 3 = T_\Pi \uparrow \omega.
\end{aligned}$$

For normal logic programs T_Π is in general not ω-continuous. Nevertheless, the least model semantics for definite programs can be used to adapt answer sets to normal programs.

Definition 2.17. *Given a normal program* Π *and an interpretation* I*, the* Gelfond-Lifschitz operator Γ_Π *is defined as follows:*

$$\Gamma_\Pi(I) = T_{\Pi/I} \uparrow \omega$$

Stable models can be defined as the fixed points of the Gelfond-Lifschitz operator Γ_Π.

Definition 2.18. *Let* Π *be a normal logic program. A two-valued interpretation* I *is a* stable model *of* Π *if and only if* $\Gamma_\Pi(I) = I$*.*

Therefore, we still have to guess the model but, then, a simple computation suffices to check whether the guess is correct.

Example 2.19. *Consider the program* Π *and the interpretations* $I_1 = \{\mathtt{b}, \mathtt{c}\}$ *and* $I_2 = \{\mathtt{a}\}$*.*

$$\mathtt{a} \leftarrow \mathtt{c}, \mathbf{not}\ \mathtt{b} \tag{2.12}$$

$$\mathtt{b} \leftarrow \mathtt{c} \tag{2.13}$$

$$\mathtt{c} \leftarrow \mathbf{not}\ \mathtt{a} \tag{2.14}$$

Π/I_1 *is obtained by removing the complete first rule, and by removing* **not** *a from the third rule.*

$$\mathtt{b} \leftarrow \mathtt{c} \tag{2.15}$$

$$\mathtt{c} \leftarrow \tag{2.16}$$

Then $T_{\Pi/I_1} \uparrow \omega = \{\mathtt{b}, \mathtt{c}\}$ *and* I_1 *is a stable model.* Π/I_2 *is*

$$\mathtt{a} \leftarrow \mathtt{c} \tag{2.17}$$

$$\mathtt{b} \leftarrow \mathtt{c} \tag{2.18}$$

We obtain $T_{\Pi/I_2} \uparrow \omega = \emptyset$*, thus* I_2 *is not a stable model.*

2.6 Well-founded Semantics

Unlike the answer set and the stable model semantics, the well-founded semantics [van Gelder et al., 1991] is based on three-valued interpretations for which the set of truth values is extended by introducing a third truth value **u** meaning undefined. This change requires that the notion of interpretation is appropriately adapted.

Definition 2.20. *A* three-valued interpretation I *of a normal program* Π *is a mapping from the Herbrand base* HB_Π *to the set of truth values* $\{\mathbf{t}, \mathbf{u}, \mathbf{f}\}$*, represented, for* $T, F \subseteq HB_\Pi$ *and* $T \cap F = \emptyset$*, by the set* (T, F)*, where elements in* T *are mapped to* $\mathbf{t}$*, elements in* F *are mapped to* $\mathbf{f}$*, and the remaining to* $\mathbf{u}$*. The set of all three-valued interpretations is denoted by* $I_{\Pi,3}$ *and each element* $(A, B) \in I_{\Pi,3}$ *can be represented using the* signed set $T \cup \mathbf{not}\, F$*.*

In the signed set notation we simply permit that negative literals are mentioned explicitly. In this way, any atom that is neither true nor false explicitly in some three-valued interpretation I is considered undefined.

The interpretation of rules can be extended to three truth values.

Definition 2.21. *Let* Π *be a normal program and* I *a three-valued interpretation of* Π*. The body of a rule* $H \leftarrow L_1, \ldots, L_n$ *in* $\mathsf{ground}(\Pi)$ *is true in* I *if and only if all literals* L_i*,* $1 \leq i \leq n$*, are true in* I*, or false in* I *if and only if at least one of that literals is false in* I*. Otherwise the body is undefined.*

The rule $H \leftarrow \mathcal{B}$ *in* $\mathsf{ground}(\Pi)$ *is true in* I *if and only if (at least) one of the following conditions holds:*

- *the head* H *is true in* I*;*
- $\mathcal{B}$ *is false in* I*;*
- $\mathcal{B}$ *is undefined and* H *is not false in* I*.*

Based on this evaluation of rules, three-valued models of normal logic programs can be defined straightforwardly.

Definition 2.22. *Let* Π *be a normal logic program. A three-valued interpretation* I *is a* three-valued model *of* Π *if and only if all rules in* $\mathsf{ground}(\Pi)$ *are true in* I*.*

Note that rules can only be true or false although some literals or even the entire head or body is undefined.

Stable models and answer sets rely on some minimality condition. Defining a similar condition for three-valued models is more difficult since there are two sets to be considered. There are two well-known orders on three-valued models, and we recall the knowledge ordering from [Fitting, 2002].

Definition 2.23. *Given two three-valued interpretations* (A, B) *and* (C, D)*, the* knowledge ordering $\leq_k$ *is defined as* $(A, B) \leq_k (C, D)$ *if and only if* $A \subseteq C$ *and* $B \subseteq D$*.*

Intuitively, this ordering minimizes everything but undefinedness, while the other ordering minimizes everything but falsity. In other words, the least model in the knowledge ordering is the one that only assigns true or false to the minimally necessary atoms.

Example 2.24. *Given* $I_1 = (\{\mathtt{a}, \mathtt{b}\}, \{\mathtt{c}\})$*,* $I_2 = (\{\mathtt{a}\}, \{\mathtt{c}, \mathtt{d}\})$*, and* $I_3 = (\{\mathtt{a}, \mathtt{b}\}, \{\mathtt{c}, \mathtt{d}\})$*, we have* $I_1 \leq_k I_3$ *and* $I_2 \leq_k I_3$ *but neither* $I_1 \leq_k I_2$ *nor* $I_2 \leq_k I_1$*. This can easily be verified in the signed set notation, because* $\{\mathtt{a}, \mathtt{b}, \mathbf{not}\ \mathtt{c}\}$ *and* $\{\mathtt{a}, \mathbf{not}\ \mathtt{c}, \mathbf{not}\ \mathtt{d}\}$ *are both subsets of* $\{\mathtt{a}, \mathtt{b}, \mathbf{not}\ \mathtt{c}, \mathbf{not}\ \mathtt{d}\}$ *but none of them is a subset of the other.*

The least three-valued model w.r.t. this order is, in fact, the well-founded model, and this model can be computed. For that, we recall the notion of an unfounded set [van Gelder et al., 1991] that is used to derive necessarily false information of a logic program given some supposed three-valued interpretation.

Definition 2.25. *Let* Π *be a normal logic program and* $I \in I_{\Pi,3}$*. We say that* $U \subseteq HB_\Pi$ *is an* unfounded set *(of* Π*)* with respect to I *if each atom* $A \in U$ *satisfies the following condition: for each rule* $A \leftarrow \mathcal{B}$ *in* $\mathsf{ground}(\Pi)$ *at least one of the following holds:*

(Ui) Some (positive or negative) literal in $\mathcal{B}$ *is false in* I*.*

(Uii) Some (non-negated) atom in $\mathcal{B}$ *occurs in* U*.*

Given a logic program Π and an interpretation $I \in I_{\Pi,3}$, there exists a greatest unfounded set of Π with respect to I, which is obtained as the union of all unfounded sets of Π with respect to I. This greatest unfounded set is used derive necessarily false information, while a generalization of the operator T_Π is used to derive necessarily true information.

Definition 2.26. *Let* Π *be a normal logic program and* I *a three-valued interpretation of* Π*. The operator* $T'_\Pi : I_{\Pi,3} \to I_{\Pi,3}$ *is defined by setting*

$$T'_\Pi(I) = \{H \mid H \leftarrow \mathcal{B} \in \mathsf{ground}(\Pi) \text{ and } \mathcal{B} \text{ is true in } I\}.$$

The derivation of necessarily true and false information is combined in the definition of the operator W_Π.

Definition 2.27. *Let* Π *be a normal logic program and* $U_\Pi(I)$ *the greatest unfounded set (of* Π*) with respect to* I*. Then for all* $I \in I_{\Pi,3}$*:*

$$W_\Pi(I) = T'_\Pi(I) \cup \mathbf{not}\, U_\Pi(I).$$

It is shown in [van Gelder et al., 1991] that W_Π is a monotonic operator and thus, by Theorem 2.2, has a least fixed point, which can be computed as usual by $W_\Pi \uparrow 0 = \emptyset$, $W_\Pi \uparrow (n+1) = W_\Pi(W_\Pi \uparrow n)$, and $W_\Pi \uparrow \alpha = \bigcup_{i<\alpha} W_\Pi \uparrow i$ for limit ordinals α.

Definition 2.28. *The* well-founded model M *of a normal logic program* Π *is obtained as the least fixed point of the operator* W_Π.

Note that W_Π is originally defined for four-valued interpretations, including a truth value 'both', which is assigned to an atom that occurs in T'_Π and U_Π. Since it is shown in [van Gelder et al., 1991] that the iteration always yields three-valued interpretations, we simplify the presentation here.

Example 2.29. *Let* Π *be a given program.*

$$\mathtt{a} \leftarrow \tag{2.19}$$

$$\mathtt{b} \leftarrow \mathbf{not}\ \mathtt{a}, \mathtt{c} \tag{2.20}$$

$$\mathtt{c} \leftarrow \mathbf{not}\ \mathtt{b} \tag{2.21}$$

$$\mathtt{d} \leftarrow \mathtt{a}, \mathbf{not}\ \mathtt{d} \tag{2.22}$$

$$\mathtt{e} \leftarrow \mathbf{not}\ \mathtt{c}, \mathtt{f} \tag{2.23}$$

$$\mathtt{f} \leftarrow \mathtt{e} \tag{2.24}$$

Now we can calculate the least fixed point of W_Π.

$$\begin{aligned}
W_\Pi \uparrow 0 &= \emptyset \\
W_\Pi \uparrow 1 &= \{\mathtt{a}, \mathbf{not}\ \mathtt{e}, \mathbf{not}\ \mathtt{f}\} \\
W_\Pi \uparrow 2 &= \{\mathtt{a}, \mathbf{not}\ \mathtt{b}, \mathbf{not}\ \mathtt{e}, \mathbf{not}\ \mathtt{f}\} \\
W_\Pi \uparrow 3 &= \{\mathtt{a}, \mathtt{c}, \mathbf{not}\ \mathtt{b}, \mathbf{not}\ \mathtt{e}, \mathbf{not}\ \mathtt{f}\} \\
W_\Pi \uparrow 4 &= \{\mathtt{a}, \mathtt{c}, \mathbf{not}\ \mathtt{b}, \mathbf{not}\ \mathtt{e}, \mathbf{not}\ \mathtt{f}\}
\end{aligned}$$

Thus $M = \{\mathtt{a}, \mathtt{c}, \mathbf{not}\ \mathtt{b}, \mathbf{not}\ \mathtt{e}, \mathbf{not}\ \mathtt{f}\}$ *is the well-founded model where* `d` *remains undefined. Note that* `e` *and* `f` *occur in the greatest unfounded set by (Uii) whereas* `b` *is contained in* U_Π *because of (Ui).*

There are several alternative definitions of the well-founded semantics. A particular one, the alternating fixed point semantics [van Gelder, 1989], establishes a close relationship with stable models. For that purpose, two sequences are defined.

Definition 2.30. *Let* Π *be a normal logic program. We define two sequences* L_i *and* G_i *as follows.*

$$\begin{array}{rrl} L_0 = \emptyset & G_0 = HB_\Pi & \\ L_{\alpha+1} = \Gamma_\Pi(L_\alpha) & G_{\alpha+1} = \Gamma_\Pi(G_\alpha) & \textit{for successor ordinal } \alpha \\ L_\alpha = \bigcup L_i & G_\alpha = \bigcap G_i & \textit{for limit ordinal } \alpha \end{array}$$

The sequence of L_i is increasing, while the sequence of G_i is decreasing. Since both sequences are monotonic, we obtain two fixed points by Theorem 2.2, a least one for the sequence of L_i and a greatest one for the sequence of G_i. For these two fixed points, the following results are shown in [van Gelder, 1989].

Theorem 2.31. *Let* Π *be a normal logic program,* L_Π *the least fixed point of the sequence of* L_i*, and* G_Π *the greatest fixed point of the Then the following hold:*

- $L_\Pi = \Gamma_\Pi(G_\Pi)$ *and* $G_\Pi = \Gamma_\Pi(L_\Pi)$*.*
- *For every stable model* I *for* Π *we have* $L_\Pi \subseteq I \subseteq G_\Pi$*.*
- $M = L_\Pi \cup \{\textbf{not}\ H \mid H \in HB_\Pi \setminus G_\Pi\}$ *is the well-founded model of* Π*.*

This important result shows that we can obtain the well-founded model by means of the operator Γ_Π. It also shows that the well-founded model approximates all stable models of a given program Π. Whatever is true in in the well-founded model, i.e., in L_Π, is also true in each stable model, and whatever is false in the well-founded model, i.e., not in G_Π, is false in each stable model of Π.

Note that this alternative definition of the well-founded model is beneficial in terms of computation. In Example 2.29, due to its simplicity, it is still considerably easy to verify that `e` and `f` form in the greatest unfounded set when computing $W_\Pi \uparrow 1$, while a (possibly automated) check for larger programs is not trivial. In contrast, computing $\Gamma_\Pi(\emptyset)$ immediately reveals that `e` and `f` are necessarily false.

These results are of particular importance for our work, since, in Chapter 6, we present a similar construction for knowledge bases consisting of a program and an ontology.

3

$\mathcal{SROIQ}$ – An Expressive Description Logic

As already mentioned in the introduction, description logics are fragments of first-order logics that are usually decidable. DLs adhere to the OWA and are commonly used for ontology languages, such as OWL, to represent taxonomic knowledge. In this chapter, we recall the necessary general notions for DLs.

Each DL, i.e., each fragment of first-order logics, can be distinguished by the admitted constructors, by means of which formulas are inductively formed. Every fragment allows a specific expressiveness, i.e., a variant of the syntax and the semantics of DLs, and algorithms that are tailored for each such fragment are defined. Here, for reasons detailed below, we opt to present[1] DL notions using the description logic $\mathcal{SROIQ}$ [Horrocks et al., 2006a]. Other description logics with different expressiveness can be easily adapted from the material shown in this chapter. For further background on Description Logics we refer to [Baader et al., 2007a] for an extensive compendium covering both introductory and advanced material on Description Logics and to [Hitzler et al., 2009b] for a book on description logics in the context of Semantic Web technologies.

$\mathcal{SROIQ}$ is the result of extending the DL $\mathcal{SHOIN}$, which is underlying the ontology language OWL DL, with expressive means that were requested by ontology developers as useful additions to OWL DL. Here, we use $\mathcal{SROIQ}$ to recall DL notions for three reasons. First, $\mathcal{SROIQ}$ is a highly expressive description logic. Thus, $\mathcal{SROIQ}$ is a very general DL containing many expressive features available for DLs.

[1] Our presentation is inspired by [Krötzsch, 2010] and [Horrocks et al., 2006a].

Second, and more importantly, $\mathcal{SROIQ}$ is the DL underlying OWL 2, the language that is the recommended standard by the W3C for modeling Semantic Web knowledge bases. Third, while the main part of our work can be considered to be applicable to any decidable first-order fragment, in Chapter 9 we study concrete cases of a top-down procedure for our developed semantics. These concrete cases are a combination of normal rules with different fragments of $\mathcal{SROIQ}$, namely those that underly the three tractable profiles of OWL 2 DL.

We present the syntax (Section 3.1) and the semantics of $\mathcal{SROIQ}$ together with the reasoning capabilities available for it (Section 3.2). Then, we also recall three fragments of $\mathcal{SROIQ}$ that underly the three tractable profiles of OWL 2 DL (Section 3.3). More precisely, we recall $\mathcal{REL}$, also known as $\mathcal{EL}^+$, which is a fragment of $\mathcal{SROEL}$ that underlies OWL EL, and its specific reasoning mechanism. Then, we present the *DL-Lite* family including *DL-Lite*$_\mathcal{R}$ that underlies OWL QL, and we finish with DLP underlying OWL RL.

3.1 Syntax of $\mathcal{SROIQ}$

Typically, formalisms founded on first-order logics are based on a signature containing the basic pieces from which the formulas are constructed. In case of description logics, such a signature consists of three disjoint sets of individual names, concept names and role names. Each of these three sets corresponds to the set of individuals, classes of individuals, and relations between classes of individuals, respectively.

Definition 3.1. *A* DL signature $\Sigma = (\mathsf{N_I}, \mathsf{N_C}, \mathsf{N_R})$ *consists of three disjoint sets of individual names* $\mathsf{N_I}$*, concept names* $\mathsf{N_C}$*, and role names* $\mathsf{N_R}$ *including the universal role* U*.*

Example 3.2. *We may consider a knowledge base describing knowledge about families, which is a commonly used example. In this case, the DL signature contains concept names, i.e., classes, such as* `Parent`*,* `Mother`*,* `Father`*, or* `Child`*, role names, such as* `hasParent`*,* `hasUncle`*, or* `IsChildOf`*, which represent relations, and individuals that may belong to the classes, such as* `Mary` *or* `John`*.*

The correspondence to signatures as defined in general (Definition 2.5) can easily be established. In first-order logic, individual names correspond to constants, concepts to unary predicates, and role names to binary predicates. The latter are of particular

interest in $\mathcal{SROIQ}$ (compared to $\mathcal{SHOIN}$) since many of the extended expressive means are related to roles. Consequently, the formalization of roles is more evolved and we start by recalling role expressions.

Definition 3.3. *Let* $\Sigma = (\mathsf{N_I}, \mathsf{N_C}, \mathsf{N_R})$ *be a DL signature. The set of* $\mathcal{SROIQ}$ role expressions $\mathbf{R}$ *is defined as*

$$\mathbf{R} = \{R \mid R \in N_R\} \cup \{R^- \mid R \in N_R\}.$$

The bijective function $\mathrm{Inv} : \mathbf{R} \to \mathbf{R}$ *is defined by setting* $\mathrm{Inv}(R) := R^-$ *and* $\mathrm{Inv}(R^-) := R$ *for all* $R \in \mathsf{N_R}$.

Role expressions simply allow us to use inverse roles directly without having to introduce concrete names for that purpose, and the function Inv simplifies that further by absorbing double inverses. Note that $\mathrm{Inv}(U) = U$.

More complex roles can be created by concatenating several roles by $\circ$, and such complex roles together with the roles in $\mathbf{R}$ can be used to define role hierarchies. Role hierarchies consist of role inclusion axioms that are formed with the help of role expressions. These hierarchies have to be restricted to regular ones to maintain decidability.

Definition 3.4. *Let* $\Sigma = (\mathsf{N_I}, \mathsf{N_C}, \mathsf{N_R})$ *be a DL signature. A generalized* role inclusion axiom *(RIA) is a statement of the form* $S_1 \circ \ldots \circ S_n \sqsubseteq R$, *where* $S_1, \ldots, S_n, R \in \mathbf{R}$. *A set of RIAs form a generalized* role hierarchy $\mathcal{R}_h$. *Such a role hierarchy* $\mathcal{R}_h$ *is* regular *if there is a strict partial order* $\prec$ *(irreflexive, asymmetric, and transitive) on* $\mathbf{R}$ *such that*

- *for* $R \notin \{S, \mathrm{Inv}(S)\}$, $S \prec R$ *iff* $\mathrm{Inv}(S) \prec R$, *and*
- *every RIA in* $\mathcal{R}_h$ *is of one of the following forms:*
  - $R \circ R \sqsubseteq R$
  - $R^- \sqsubseteq R$
  - $S_1 \circ \ldots \circ S_n \sqsubseteq R$
  - $R \circ S_1 \circ \ldots \circ S_n \sqsubseteq R$
  - $S_1 \circ \ldots \circ S_n \circ R \sqsubseteq R$

  such that $R, S_1, \ldots, S_n$ *are pairwise different roles in* $\mathbf{R}$, *and* $S_i \prec R$ *for* $i = 1, \ldots, n$.

The restrictions imposed essentially ensure that there are no cyclic dependencies for different roles (either directly or by transitivity). In fact, cyclic dependencies are limited to express transitivity, symmetry, and right and left identity of some role. Other than that, the roles on the left hand side of every RIA have to be smaller w.r.t. the order.

Example 3.5. *The next two examples have been taken from [Motik et al., 2009b]. Consider at first that the wife of one's uncle is one's aunt in law. This relation can be expressed with the two RIAs.*

$$\texttt{HasFather} \circ \texttt{HasBrother} \sqsubseteq \texttt{HasUncle} \tag{3.1}$$

$$\texttt{HasUncle} \circ \texttt{HasWife} \sqsubseteq \texttt{HasAuntInLaw} \tag{3.2}$$

It is easy to see that these two RIAs form a regular role hierarchy. We can, e.g., consider the order of roles as follows:

$$\texttt{HasFather} \prec \texttt{HasBrother} \prec \texttt{HasUncle} \prec \texttt{HasWife} \prec \texttt{HasAuntInLaw}$$

As an example for a hierarchy that is not regular, consider the information that the uncle of one's child is one's brother. This relation can also be expressed with the two RIAs.

$$\texttt{HasFather} \circ \texttt{HasBrother} \sqsubseteq \texttt{HasUncle} \tag{3.3}$$

$$\texttt{HasChild} \circ \texttt{HasUncle} \sqsubseteq \texttt{HasBrother} \tag{3.4}$$

In this case it is impossible to obtain a strict order. We always obtain that `HasBrother` *is below itself.*

$$\texttt{HasFather} \prec \texttt{HasBrother} \prec \texttt{HasUncle} \prec \texttt{HasChild} \prec \texttt{HasBrother}$$

We can express additional similar properties, such as reflexivity or role-disjointness, but, for that purpose, we need to introduce a further syntactic restriction on roles to maintain decidability, namely simple roles.

Definition 3.6. *Let $\Sigma = (\mathsf{N_I}, \mathsf{N_C}, \mathsf{N_R})$ be a DL signature and $\mathcal{R}_h$ a role hierarchy. The set of* simple roles *for $\mathcal{R}_h$ is defined inductively as follows:*

- *If R occurs on the right hand side of an RIA, then R is simple if, for each $w \sqsubseteq R \in \mathcal{R}_h$, $w = S$ for a simple role S.*

- *The inverse of a simple role is simple.*

The first item includes the special case that R does not appear on the right hand side of any RIA.

Example 3.7. *We add another example taken from [Motik et al., 2009b].*

$$\texttt{HasFather} \circ \texttt{HasBrother} \sqsubseteq \texttt{HasUncle} \tag{3.5}$$

$$\texttt{HasUncle} \sqsubseteq \texttt{HasRelative} \tag{3.6}$$

$$\texttt{HasBiologicalFather} \sqsubseteq \texttt{HasFather} \tag{3.7}$$

`HasBiologicalFather` *does not appear on the right hand side of any RIA. Thus, this role and* `HasFather` *are both simple. In opposite to that,* `HasUncle` *appears on the right hand side of a non-simple RIA, so* `HasUncle` *and* `HasRelative` *are both not simple roles.*

Now we are able to define the $\mathcal{SROIQ}$ RBox as the combination of a regular role hierarchy and set of role assertions,[1] which expresses relations on roles.

Definition 3.8. *Given a DL signature* $\Sigma = (\mathsf{N_I}, \mathsf{N_C}, \mathsf{N_R})$*, a* role assertion *is a statement of the form:* $\mathsf{Ref}(R)$ *(reflexivity),* $\mathsf{Tra}(R)$ *(transitivity),* $\mathsf{Irr}(S)$ *(irreflexivity),* $\mathsf{Dis}(S,T)$ *(role disjointness),* $\mathsf{Sym}(R)$ *(symmetry),* $\mathsf{Asy}(S)$ *(asymmetry), where* $R, S, T \in \mathbf{R}$ *and* S *and* T *are simple. A* $\mathcal{SROIQ}$ RBox $\mathcal{R}$ *is the union of a set of role assertions together with a regular role hierarchy.*

It should be pointed out that there is an alternative way of declaring simple and non-simple role names a priori. This may be advantageous for the investigation of theoretical properties. But since we are not going to investigate theoretical properties of description logics alone, we use the definition that is more reasonable in practice if we consider dynamic ontologies. This also shows that the increased expressiveness compared, e.g., to $\mathcal{SHOIN}$ results in a more costly ontology maintenance.

Given an RBox, we can also define complex concept expressions.

Definition 3.9. *Let* $\Sigma = (\mathsf{N_I}, \mathsf{N_C}, \mathsf{N_R})$ *be a DL signature and* $\mathcal{R}$ *a* $\mathcal{SROIQ}$ *RBox. The set of* concept expressions *(or simply* concepts*)* $\mathbf{C}$ *is defined as follows:*

- $\mathsf{N_C} \subseteq \mathbf{C}$, $\top \in \mathbf{C}$, $\bot \in \mathbf{C}$

[1] Some of the role assertions are syntactic sugar since they can be expressed directly with an RIA, but we present them anyway, for convenience.

- *if* $C, D \in \mathbf{C}$, $R, S \in \mathbf{R}$, S *a simple role,* $a \in \mathsf{N_I}$, *and* n *a non-negative integer, then* $\neg C$, $C \sqcap D$, $C \sqcup D$, $\{a\}$, $\forall R.C$, $\exists R.C$, $\exists S.\mathsf{Self}$, $\leqslant nS.C$, *and* $\geqslant nS.C$ *are also concept expressions.*

The complex expressions in the second item are called, respectively, (classical) negation, conjunction, disjunction, nominal, universal restriction, existential restriction, Self concept, and qualified number restrictions, which are divided into at-most and at-least restriction.

Example 3.10. *These constructors allow us to express, e.g., the set of all individuals that have at least two children that are female (see (1.1) in Chapter 1):*

$$(\geqslant 2\,\mathtt{HasChild}) \sqcap (\forall\,\mathtt{HasChild}.\mathtt{Female}) \tag{3.8}$$

We may also represent the set of all individuals that are grandparents or have no sons.

$$(\exists\,\mathtt{HasChild}.\exists\,\mathtt{HasCild}.\top) \sqcup (\forall\,\mathtt{HasChild}.\neg\mathtt{Male}) \tag{3.9}$$

The concept Self *can be used to express reflexivity in concepts, such as 'narcist' using* $\exists\,\mathtt{Likes}.\mathsf{Self}$, *while nominals are generalizations of individuals in the sense that we can express the class of all individuals that have* John *as a child:* $\exists\,\mathtt{HasChild}.\{\mathtt{John}\}$.

These concept expressions together with the RBox in consideration enable us to define the notions of a TBox (to express relations between concepts), an ABox (to assert properties on individuals), and the combination of all axioms for roles, concepts, and constants, i.e., a $\mathcal{SROIQ}$ knowledge base.

Definition 3.11. *Let* $\Sigma = (\mathsf{N_I}, \mathsf{N_C}, \mathsf{N_R})$ *be a DL signature,* $\mathcal{R}$ *a* $\mathcal{SROIQ}$ *RBox,* $a, b \in N_I$ *individual names,* $C, D \in \mathbf{C}$ *concept expressions, and* $R \in \mathbf{R}$ *a role expression. We define a* $\mathcal{SROIQ}$ TBox $\mathcal{T}$ *for* $\mathcal{R}$ *as a set of* general concept inclusion axioms *(GCIs) of the form* $C \sqsubseteq D$. *A* $\mathcal{SROIQ}$ ABox $\mathcal{A}$ *is a set of* individual assertions *of one of the following forms:* $C(a)$, $R(a, b)$, $\neg R(a, b)$, *and* $a \not\approx b$. *A* $\mathcal{SROIQ}$ *knowledge base* $\mathcal{O}$ *is the union of a regular RBox* $\mathcal{R}$, *a TBox* $\mathcal{T}$ *for* $\mathcal{R}$, *and an ABox* $\mathcal{A}$.

We use the notation $\mathcal{O}$ for any DL knowledge base, even if it does not have the all the three possible components.

3.2 Semantics of $\mathcal{SROIQ}$

Given a $\mathcal{SROIQ}$ knowledge base, its semantics is provided in the usual way applied in Description Logics, i.e., a first-order model theory is used. First, an interpretation is defined that contains the domain and interprets the elements of the DL signature.

Definition 3.12. *Given DL signature* $\Sigma = (\mathsf{N_I}, \mathsf{N_C}, \mathsf{N_R})$*, an* interpretation $\mathcal{I}$ *is a pair* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ *consisting of a non-empty set* $\Delta^{\mathcal{I}}$ *and a mapping* $\cdot^{\mathcal{I}}$ *that ensures that*

- $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ *for each* $a \in \mathsf{N_I}$*;*
- $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ *for each* $C \in \mathsf{N_C}$*; and*
- $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ *for each* $R \in \mathsf{N_R}$*.*

This mapping can be extended to arbitrary role and concept expressions as shown in Fig. 3.1.

Name	Syntax	Semantics
inverse role	R^-	$\{(x,y) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (y,x) \in R^{\mathcal{I}}\}$
universal role	U	$\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
top	$\top$	$\Delta^{\mathcal{I}}$
bottom	$\bot$	$\emptyset$
negation	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
disjunction	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
nominals	$\{a\}$	$\{a^{\mathcal{I}}\}$
universal restriction	$\forall R.C$	$\{x \in \Delta^{\mathcal{I}} \mid (x,y) \in R^{\mathcal{I}} \text{ implies } y \in C^{\mathcal{I}}\}$
existential restriction	$\exists R.C$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y \in \Delta^{\mathcal{I}}, (x,y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
Self concept	$\exists S.\mathsf{Self}$	$\{x \in \Delta^{\mathcal{I}} \mid (x,x) \in S^{\mathcal{I}}\}$
qualified number	$\leqslant nS.C$	$\{x \in \Delta^{\mathcal{I}} \mid \sharp\{(x,y) \in S^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\} \leq n\}$
restrictions	$\geqslant nS.C$	$\{x \in \Delta^{\mathcal{I}} \mid \sharp\{(x,y) \in S^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\} \geq n\}$

Figure 3.1: Semantics of role and concept expressions in $\mathcal{SROIQ}$ for an interpretation $\mathcal{I}$ with domain $\Delta^{\mathcal{I}}$

We note that if the DL signature is clear from the context, then we do not mention it explicitly.

Next, we define when an interpretation is a model for some DL axiom.

Definition 3.13. *Given an interpretation $\mathcal{I}$ and a $\mathcal{SROIQ}$ (RBox, TBox, or ABox) axiom α, we say that $\mathcal{I}$* satisfies *α, written $\mathcal{I} \models \alpha$, if the respective conditions of Fig. 3.2 are satisfied. $\mathcal{I}$* satisfies *a $\mathcal{SROIQ}$ knowledge base $\mathcal{O}$, denoted as $\mathcal{I} \models \mathcal{O}$, if it satisfies all of its axioms. We also say that $\mathcal{I}$ is a* model *of a given axiom, and $\mathcal{I}$ is a model of the knowledge base $\mathcal{O}$.*

Note, that $\circ$ on the right hand side of Fig. 3.2 denotes standard composition of binary relations, i.e., $R^{\mathcal{I}} \circ S^{\mathcal{I}} = \{(x, z) \mid (x, y) \in R^{\mathcal{I}} \wedge (y, z) \in S^{\mathcal{I}}\}$ holds.

Axiom α	Condition for $\mathcal{I} \models \alpha$
$R_1 \circ \ldots \circ R_n \sqsubseteq R$	$R_1^{\mathcal{I}} \circ \ldots \circ R_n^{\mathcal{I}} \sqsubseteq R^{\mathcal{I}}$
$\mathsf{Tra(R)}$	if $R^{\mathcal{I}} \circ R^{\mathcal{I}} \subseteq R^{\mathcal{I}}$
$\mathsf{Ref(R)}$	$(x, x) \in R^{\mathcal{I}}$ for all $x \in \Delta^{\mathcal{I}}$
$\mathsf{Irr(S)}$	$(x, x) \notin S^{\mathcal{I}}$ for all $x \in \Delta^{\mathcal{I}}$
$\mathsf{Dis(S, T)}$	if $(x, y) \in S^{\mathcal{I}}$ then $(x, y) \notin T^{\mathcal{I}}$ for all $x, y\Delta^{\mathcal{I}}$
$\mathsf{Sym(R)}$	if $(x, y) \in R^{\mathcal{I}}$ then $(y, x) \in R^{\mathcal{I}}$ for all $x, y \in \Delta^{\mathcal{I}}$
$\mathsf{Asy(S)}$	$(x, y) \in S^{\mathcal{I}}$ then $(y, x) \notin S^{\mathcal{I}}$ for all $x, y \in \Delta^{\mathcal{I}}$
$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
$C(a)$	$a^{\mathcal{I}} \in C^{\mathcal{I}}$
$R(a, b)$	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$
$\neg R(a, b)$	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \notin R^{\mathcal{I}}$
$a \not\approx b$	$a^{\mathcal{I}} \neq b^{\mathcal{I}}$

Figure 3.2: Semantics of $\mathcal{SROIQ}$ axioms for an interpretation $\mathcal{I}$ with domain $\Delta^{\mathcal{I}}$

We are now able to derive model-theoretic notions for $\mathcal{SROIQ}$.

Definition 3.14. *Let $\mathcal{O}$ and $\mathcal{O}'$ be $\mathcal{SROIQ}$ knowledge bases. We define that*

- *$\mathcal{O}$ is* satisfiable *(consistent) if it has a model and* unsatisfiable *(inconsistent) otherwise,*
- *$\mathcal{O}$* entails *$\mathcal{O}'$, written $\mathcal{O} \models \mathcal{O}'$, if all models of $\mathcal{O}$ are also models of $\mathcal{O}'$.*

This terminology is extended to axioms by treating them as singleton knowledge bases. A knowledge base or axiom that is entailed is also called a logical consequence.

It should be pointed out that various properties of first-order logics hold immediately also for DLs , $\mathcal{SROIQ}$. For example, DLs are monotonic, i.e., adding information never

reduces the amount of logical consequences - the addition of further axioms decreases the amount models and thus augments the set of logical consequences. Moreover, if a $\mathcal{SROIQ}$ knowledge base is inconsistent, then it allows us to derive all possible logical consequences.

Next, we recall the reasoning tasks (from Krötzsch [2010]) that are considered standard for deriving logical consequences from DL knowledge bases.

Definition 3.15. *Consider a $\mathcal{SROIQ}$ knowledge base $\mathcal{O}$. The* standard reasoning tasks *of description logics are described as follows.*

- Inconsistency checking*: Is $\mathcal{O}$ inconsistent?*
- Concept subsumption*: Given concepts C, D, does $\mathcal{O} \models C \sqsubseteq D$ hold?*
- Instance checking*: Given a concept C and an individual name a, does $\mathcal{O} \models C(a)$ hold?*
- Concept unsatisfiability*: Given a concept C, is it the case that there is no model $\mathcal{I} \models \mathcal{O}$ such that $C^{\mathcal{I}} \neq \emptyset$?*

It is well known that these standard reasoning tasks can be reduced to each other in linear time (recalled from [Krötzsch, 2010]).

Proposition 3.16. *Let $\mathcal{O}$ be a $\mathcal{SROIQ}$ knowledge base. The standard reasoning tasks in $\mathcal{SROIQ}$ can be reduced to each other in linear time, and this is possible in any fragment of $\mathcal{SROIQ}$ that includes axioms of the form $A(a)$ and $A \sqcap C \sqsubseteq \bot$.*

Proof. We find that $\mathcal{O}$ is inconsistent if the concept $\top$ is unsatisfiable. C is unsatisfiable in $\mathcal{O}$ if $\mathcal{O} \models C \sqsubseteq \bot$. Given a new individual name a, we obtain $\mathcal{O} \models C \sqsubseteq D$ if $\mathcal{O} \cup \{C(a)\} \models D(a)$. For a new concept name A, $\models C(a)$ if $\mathcal{O} \cup \{A(a), A \sqcap C \sqsubseteq \bot\}$ is inconsistent. This cyclic reduction shows that all reasoning problems can be reduced to one another. □

There are other reasoning tasks that are sometimes considered as standard as well, such as instance retrieval, i.e., finding all instances of a concept, or classification, i.e., computing all subsumptions between all concept names in the knowledge base. However, these are no decision problems and, in general, are not of the same worst case complexity as the reasoning tasks presented in Definition 3.15. This is why we do not include them in the list of standard reasoning tasks.

Besides that, several other reasoning tasks have been studied in Description Logics. Among the examples are the computation of explanations for logical entailments [Horridge et al., 2008; Kalyanpur, 2006], and least common subsumer concepts that generalize given concept expressions in description logics where union of concepts is not available [Baader, 2003; Baader et al., 2007b]. Another relevant inference task is conjunctive query answering [Calvanese et al., 1998; Glimm et al., 2008; Lutz, 2008].

Definition 3.17. *Let $\vec{x}$ and $\vec{y}$ be disjoint vectors of* distinguished *and* non-distinguished *variables, respectively. A* conjunctive query $Q(\vec{x})$ *is a finite formula of the form $\exists \vec{y} : A_1 \wedge \ldots \wedge A_n$ where each A_i is a function-free first-order atom over predicates and constants from Σ and the variables from $\vec{x} \cup \vec{y}$. If $\vec{x}$ is empty, the conjunctive query is called* Boolean. *Given a vector $\vec{a}$ of as a many constants as there are variables in $\vec{x}$, the result of replacing in $Q(\vec{x})$ each $x_i \in \vec{x}$ with the corresponding $a_i \in \vec{a}$ is denoted $Q(\vec{a})$. We say that $\vec{a}$ is an* answer *to $Q(\vec{x})$ over a DL knowledge base $\mathcal{O}$, written $\mathcal{O} \models Q(\vec{a})$, if $Q(\vec{a})$ is true in every model of $\mathcal{O}$. The problem of checking whether $\vec{a}$ is a answer to $Q(\vec{x})$ is called* query answering.

As shown in [Kazakov, 2008], standard reasoning in $\mathcal{SROIQ}$ is N2ExpTime complete. This high computational complexity is justified by the expressiveness of the language in consideration.

We finish this section with a (general) example on DL knowledge bases in the context of the previously introduced example on recommendations for CDs.

Example 3.18. *Consider the online store scenario of Example 1.1. The following axioms and assertions could be part of the ontology for the store:*[1]

$$\texttt{CD} \sqsubseteq \exists \texttt{HasPiece}.\texttt{Piece} \tag{3.10}$$

$$\texttt{Piece} \sqsubseteq \exists \texttt{HasArtist}.\texttt{Artist} \tag{3.11}$$

$$\texttt{HasPiece} \circ \texttt{HasArtist} \sqsubseteq \texttt{HasArtist} \tag{3.12}$$

$$\texttt{TopSeller} \sqcup \texttt{SpecialOffer} \sqsubseteq \texttt{Recommend} \tag{3.13}$$

$$\texttt{HasPiece}(\texttt{BNAW}, \texttt{BlueTrain}) \tag{3.14}$$

$$\texttt{HasArtist}(\texttt{BlueTrain}, \texttt{JohnColtrane}) \tag{3.15}$$

[1] In these axioms, and throughout the thesis, we adopt the convention that names starting with a capital letter represent concepts and roles (termed DL-atoms) and objects/individuals, while names starting with a lower case letter represent variables and predicates not appearing in the ontology (called non-DL atoms).

Axiom (3.10) states that each CD consists of at least one piece and axiom (3.11) expresses that each piece of music has an artist. The role composition axiom (3.12) states that if x is related to y by `HasPiece` *and y is related to z by* `HasArtist` *then x is related to z by* `HasArtist`*, i.e.,* `HasArtist` *is a left-identity role. (3.10) - (3.12) alone enable us to derive, e.g., that the artist of a piece on a certain CD is an artist of that CD. Note that this conclusion can be drawn without any present CDs, artists or pieces of music, as intended when reasoning with schema knowledge in an infinite domain. Of course, once specific information is available (assertions 3.14 and 3.15), we are able to derive, e.g., that* `JohnColtrane` *is an artist of the album* `BNAW`*, and likewise for all the other artists on that CD that are not explicitly mentioned.*

Axiom (3.13) expresses one general guideline for recommendations: CDs that are special offers or top sellers are automatically recommended to the customers.

3.3 Tractable Fragments of $\mathcal{SROIQ}$

Since in some applications not all the constructors that are available in $\mathcal{SROIQ}$ are actually needed, we can consider fragments of $\mathcal{SROIQ}$ and rely on computationally cheaper algorithms. This has been done for OWL 2 DL (and its underlying DL $\mathcal{SROIQ}$) and three tractable profiles of OWL 2 were introduced [Motik et al., 2009a]. Each of these tractable profiles corresponds to some concrete DL, and, in this section, we recall briefly the DLs that underly the three tractable profiles of OWL 2.

3.3.1 $\mathcal{REL}$

The first tractable fragment is $\mathcal{SROEL}$, also known as $\mathcal{EL}^{++}$ [Baader et al., 2005, 2008], which underlies OWL 2 EL. Here, we limit ourselves to the fragment $\mathcal{REL}$, also known as $\mathcal{EL}^{+}$,[1] i.e., we avoid concrete domains as before in the case of $\mathcal{SROIQ}$ and also nominals and complex role expressions, such as reflexivity.

[1] Strictly speaking we present $\mathcal{EL}^{+}$ augmented with $\bot$. Note that $\mathcal{EL}^{++}$ and $\mathcal{EL}^{+}$ are historically the commonly used names. However, the usage of + or ++ is rather imprecise in the sense that it does not hint on the used constructors, whereas the adaptation of the general nomenclature of DLs in Section 3.5 of [Krötzsch, 2010] to $\mathcal{EL}$ languages more clearly designates the appearing constructors. In fact, in case of $\mathcal{SROEL}$, $\mathcal{O}$ does refer to the occurrence of nominals and $\mathcal{S}$ indicates the availability of, e.g., reflexivity. The language $\mathcal{REL}$ lacks these features and this can be witnessed by the absence of the corresponding letters in the name of the DL. Due to this gain in clarity we also adopt this nomenclature.

Name	Syntax	Semantics
top	$\top$	$\Delta^{\mathcal{I}}$
bottom	$\bot$	$\emptyset$
conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
existential restr.	$\exists R.C$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y \in \Delta^{\mathcal{I}} : (x,y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
GCI	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
RIA	$R_1 \circ \cdots \circ R_k \sqsubseteq R$	$R_1^{\mathcal{I}} \circ \cdots \circ R_k^{\mathcal{I}} \sqsubseteq R^{\mathcal{I}}$
individual assertion	$C(a)$	$a^{\mathcal{I}} \in C^{\mathcal{I}}$
individual assertion	$R(a,b)$	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$

Figure 3.3: Syntax and semantics of $\mathcal{REL}$.

For simplicity we present in Figure 3.3 the syntax and semantics of all constructors/axioms allowed in $\mathcal{REL}$. The remainder can then be straightforwardly adapted from the material presented on $\mathcal{SROIQ}$. We just like to point out a few things. First, we can simply ignore simple roles here since none of the axioms involving simple roles are allowed in $\mathcal{REL}$. This includes role assertions expressing properties, such as role disjointness or symmetry. Nevertheless, some of these role assertions, such as transitivity, can be expressed via RIAs directly, i.e., $\mathsf{Tra}(\mathsf{R})$ for some role R can be represented by $R \circ R \sqsubseteq R$ as pointed out in Section 3.1. Likewise, $\mathcal{SROEL}$ does not impose any requirements on the regularity of role hierarchies, which also means that full $\mathcal{REL}$ is strictly speaking not a fragment of $\mathcal{SROIQ}$. However, conjunctive query answering for unrestricted $\mathcal{SROEL}$ becomes undecidable [Rosati, 2007], and only the restriction to regular role hierarchies ensures decidability [Krötzsch et al., 2007; Mei et al., 2009]. We therefore assume that the role hierarchy is regular even though this is not a requirement of $\mathcal{REL}$ in general. Finally, the usage of $\bot$ allows us to express disjointness and unsatisfiability of concepts even though negation is not available in $\mathcal{REL}$.

Standard reasoning in $\mathcal{SROEL}$ is PTIME-complete while answering conjunctive queries (restricted to regular RBoxes) is PSPACE-complete [Krötzsch et al., 2007]. This results carry over to $\mathcal{REL}$.

$\mathcal{SROEL}$ and its corresponding OWL 2 Profile OWL 2 EL are mainly used in applications where there is a large amount of concepts and relations between them, i.e., knowledge bases with a large TBoxes and RBoxes. Examples for that are SNOMED

or GALEN.

We would like to point out that Example 3.18 is of course a $\mathcal{SROIQ}$ knowledge base. However, if we split 3.13 into two statements, then the knowledge base is also in $\mathcal{REL}$.

3.3.2 *DL-Lite*$_\mathcal{R}$

While OWL 2 EL is used for applications where reasoning on large TBoxes and RBoxes is required, another profile, namely OWL 2 QL, is used if we are more interested in efficient querying.

The description logic underlying OWL 2 QL is *DL-Lite*$_\mathcal{R}$, one language of the *DL-Lite* family [Calvanese et al., 2007]. We follow the presentation of [Calvanese et al., 2007] and recall the *DL-Lite*$_\mathcal{R}$ language.

Unlike $\mathcal{REL}$, *DL-Lite*$_\mathcal{R}$ not only is limited to certain constructors but also restricts what kind of constructors may appear on the left and the right hand side of GCIs. The TBox contains only GCIs whose left and right hand sides (subclass and superclass) are constructed w.r.t. the restrictions presented in Fig. 3.4. Note that $\exists R.\top$ is a limited form of existential restriction.

subclass expressions	superclass expressions
$C \in \mathsf{N_C}$	$C \in \mathsf{N_C}$
$\exists R.\top$	$\exists R.D$
$\sqcup$	$\sqcap$
$\bot$	$\top$
	$\neg$

Figure 3.4: Syntactic restrictions on class expressions in *DL-Lite*$_\mathcal{R}$ where D is any allowed concept expression and $R \in \mathcal{R}$.

It is explained in [Calvanese et al., 2007] that $\sqcup$ on the left hand side and $\sqcap$ on the right hand are syntactic sugar in the sense that these can alternatively be represented by two statements. For example, $C_1 \sqcup C_2 \sqsubseteq D$ is equivalently to $C_1 \sqsubseteq D$ and $C_2 \sqsubseteq D$. Likewise, $\bot$ on the left hand side and $\top$ on the right hand side are superfluous.

ABox statements are limited to $A(a)$ and $R(a, b)$ where $A \in \mathsf{N_C}$ and $R \in \mathsf{N_R}$. Assertions $C(a)$ for arbitrary concepts C can be simulated by adding $A \sqsubseteq C$ to the

TBox and $A(a)$ to the ABox.

The semantics can be derived from the material presented on $\mathcal{SROIQ}$ with one exception: here it is assumed that to each constant $a \in \mathsf{N_I}$ a distinct object $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ is assigned. In other words, in *DL-Lite*$_{\mathcal{R}}$ the unique name assumption is applied and it is shown in [Artale et al., 2009] that dropping the unique name assumption would increase significantly the computational complexity.

Additionally, *DL-Lite*$_{\mathcal{R}}$ allows us to add an RBox containing one role $R \in \mathcal{R}$ on the left hand side and one (possibly negated) role $S \in \mathcal{R}$ on the right hand side. Such axioms also permit to express properties, such as symmetry.

Standard reasoning tasks in *DL-Lite*$_{\mathcal{R}}$ are polynomial in the size of the TBox and the RBox, and in LogSpace in the size of the ABox, i.e., in data complexity. The same holds for answering conjunctive queries, but if we consider the combined knowledge base, then answering conjunctive queries is NP-complete.

Note that the knowledge base in Example 3.18 is also in *DL-Lite*$_{\mathcal{R}}$.

3.3.3 DLP

The third fragment of $\mathcal{SROIQ}$, DLP [Grosof et al., 2003], is quite closely related to rules as presented in Chapter 2, and it is the DL underlying OWL 2 RL. Essentially, DLP is the translation of Description Horn Logic (DLH) into rules, where DLH corresponds to the fragment of $\mathcal{SROIQ}$ that is translatable into definite Horn rules, i.e., rules without negation and without function symbols. The restrictions on GCIs are caused by the expressive limitations of rules and are specified in Fig. 3.5 for the left and right hand side of GCIs.

subclass expressions	superclass expressions
$C \in \mathsf{N_C}$	$C \in \mathsf{N_C}$
$\sqcap$	$\sqcap$
$\exists R.C$	$\forall R.C$
$\sqcup$	

Figure 3.5: Syntactic restrictions on class expressions in DLP where C is any allowed concept expression and $R \in \mathcal{R}$.

Note that $\sqcup$ on the left hand side and $\sqcap$ on the right hand are, yet again, syntactic sugar in the sense that these can alternatively be represented by two statements.

DLH also allows the usage of role inclusion axioms and therefore straightforwardly the role assertions $\mathsf{Tra}(R)$ and $\mathsf{Sym}(R)$ for a role $R \in \mathcal{R}$. Finally, assertions of the form $C(a)$ for $C \in \mathsf{N_C}$ and $R(a, b)$ for $R \in \mathcal{R}$ are allowed as axioms in the ABox. More complex assertions where C is a concept name can be derived by introducing appropriate GCIs. For example, $(\forall R.D)(a)$ can be expressed by $C(a)$ and $C \sqsubseteq \forall R.D$.

Given these restrictions, it can be shown that the standard reasoning tasks are polynomial in the size of the TBox. It should be noted that satisfiability checking is irrelevant for DLP due to the absence of negation.

Finally, we want to mention that Example 3.18 is not in DLP due the presence of $\exists$ on the right hand side of two GCIs.

4

MKNF Logics and Hybrid MKNF Knowledge Bases

In Chapter 1, we have shown that there is a large number of different approaches for combining rules and ontologies as presented in the previous two chapters, and we pointed out that the approach on which we base our work is hybrid MKNF knowledge bases [Motik and Rosati, 2010]. This work is based on the logic of minimal knowledge and negation as failure (MKNF) [Lifschitz, 1991] and is closely related to the logic of minimal belief and negation as failure (MBNF) [Lifschitz, 1994]. Both were defined as unifying frameworks for formalisms that include non-monotonic such as rules in logic programs, default logics [Reiter, 1980] or autoepistemic logics [Moore, 1985].

Here, following the presentation in [Motik and Rosati, 2010], we are going to recall the syntax (Section 4.1) and the semantics of MKNF logics (Section 4.2). Then, we recall an important change to the semantics, i.e. the standard names assumption (Section 4.3), before we present the most general form of combination of rules and ontologies in that framework – MKNF$^+$ knowledge bases (Section 4.4). We continue this chapter with some considerations on semantics of such knowledge bases including how to transform them into MKNF knowledge bases (Section 4.5), and we present the restrictions that are applied such that reasoning in hybrid MKNF becomes decidable (Section 4.6). Moreover, we define specific restrictions for our work and justify them (Section 4.7).

4.1 Syntax of MKNF Logics

The logic of minimal knowledge and negation as failure extends first-order logic with two modal operators **K** and **not** that inspect the knowledge base: intuitively, given a first-order formula φ, $\mathbf{K}\,\varphi$ asks whether φ is known (or derivable as a logical consequence), while $\mathbf{not}\,\varphi$ is used to check whether φ is not known. Thus, the two modal operators permit local closed world reasoning by inspecting the knowledge base looking for derivable information. In particular, the operator **not** allows one to draw conclusions from the absence of information, in a way similar to that of default negation in Logic Programming.

Now, we present the formal syntax of MKNF logics as introduced in [Motik and Rosati, 2007, 2010].

Definition 4.1. *Let $\Sigma = (\Sigma_c, \Sigma_f, \Sigma_p)$ be a signature and Σ_p contain the binary equality predicate $\approx$. The syntax of MKNF formulas over Σ is defined by the following grammar, where t_i are first-order terms and P is a predicate.*

$$\varphi \leftarrow P(t_1, \ldots, t_n) \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \exists x.\varphi \mid \mathbf{K}\,\varphi \mid \mathbf{not}\,\varphi \tag{4.1}$$

Moreover, $\varphi_1 \vee \varphi_2$, $\varphi_1 \supset \varphi_2$, $\varphi_1 \equiv \varphi_2$, $\forall x : \varphi$, $\mathbf{t}$, $\mathbf{f}$, $t_1 \approx t_2$, and $t_1 \not\approx t_2$ are abbreviations, respectively, for $\neg(\neg\varphi_1 \wedge \neg\varphi_2)$, $\neg\varphi_1 \vee \varphi_2$, $(\varphi_1 \supset \varphi_2) \wedge (\varphi_2 \supset \varphi_1)$, $\neg(\exists x : \neg\varphi)$, $a \vee \neg a$, $a \wedge \neg a$, $\approx (t_1, t_2)$, and $\neg(t_1 \approx t_2)$.

First-order atoms of the form $t_1 \approx t_2$ (resp. $t_1 \not\approx t_2$) are called equalities *(resp.* inequalities*), and $\varphi[t_1/x_1, \ldots, t_n/x_n]$ denotes the formula obtained by substituting the free variables x_i in φ, i.e., the variables that are not in the scope of any quantifier, by the terms t_i.*

Given a (first-order) formula φ, $\mathbf{K}\,\varphi$ is called a **K**-atom *and* $\mathbf{not}\,\varphi$ *a* **not**-atom*;* **K**-*atoms and* **not**-*atoms are* modal atoms*.*

Given this grammar[1], we can form different MKNF formulas:

Definition 4.2. *An MKNF formula φ is called* strict*, if there is no modal atom in φ that occurs in the scope of a modal operator. An MKNF formula φ without any free variables is* closed*, and an MKNF formula φ is* ground *if φ does not contain variables at all. An MKNF formula φ is* modally closed *if all modal operators (***K** *and* **not***) are applied in φ only to closed subformulas, and φ is* positive *if φ does not contain the*

[1] To be precise, the transitions $\varphi_1 \leftarrow \varphi$ and $\varphi_2 \leftarrow \varphi$ are implicitly required.

operator **not**. *An MKNF formula φ is* subjective *if all first-order atoms of φ occur within the scope of a modal operator, and φ is* flat *if φ is subjective and all occurrences of modal atoms in φ are strict.*

4.2 Semantics of MKNF Logics

The semantics of MKNF logics is based on first-order interpretations, though a slightly different version is used here.

Definition 4.3. *Let Σ be a signature and Δ a universe such that, for each element $\alpha \in \Delta$, Σ is required to contain a special constant n_α – called a* name *– such that $n_\alpha^I = \alpha$. A first-order interpretation I over Σ and Δ assigns an object $a^I \in \Delta$ to each constant $a \in \Sigma_c$, a function $f^I : \Delta^n \to \Delta$ to each n-ary function symbol $f \in \Sigma_f$, and a relation $P^I \subseteq \Delta^n$ to each n-ary predicate $P \in \Sigma_p$. The predicate $\approx$ is interpreted in I as equality, i.e., for $\alpha, \beta \in \Delta$, we have $(\alpha, \beta) \in \approx^I$ iff $\alpha = \beta$.*

The interpretation of a variable-free term $t = f(s_1, \ldots, s_n)$ is defined recursively as $t^I = f^I(s_1^I, \ldots, s_n^I)$.

The semantics of an MKNF formula over a signature Σ (henceforth considered implicit in all definitions) is based on a specific structure and defined as follows.

Definition 4.4. *An* MKNF structure *is a triple (I, M, N) where I is a first-order interpretation over Δ and Σ, and M and N are nonempty sets of first-order interpretations over Δ and Σ. Given an MKNF structure (I, M, N), satisfiability of closed MKNF formulas is defined as follows:*

$$
\begin{array}{ll}
(I,M,N) \models P(t_1,\ldots,t_n) & \text{iff } (t_1^I,\ldots,t_n^I) \in P^I \\
(I,M,N) \models \neg\varphi & \text{iff } (I,M,N) \not\models \varphi \\
(I,M,N) \models \varphi_1 \wedge \varphi_2 & \text{iff } (I,M,N) \models \varphi_1 \text{ and } (I,M,N) \models \varphi_2 \\
(I,M,N) \models \exists x : \varphi & \text{iff } (I,M,N) \models \varphi[n_\alpha/x] \text{ for some } \alpha \in \Delta \\
(I,M,N) \models \mathbf{K}\,\varphi & \text{iff } (J,M,N) \models \varphi \text{ for all } J \in M \\
(I,M,N) \models \mathbf{not}\,\varphi & \text{iff } (J,M,N) \not\models \varphi \text{ for some } J \in N
\end{array}
$$

Note that the evaluation of **K** and **not** are kept separate in this definition of satisfiability.

Example 4.5. *Consider an MKNF structure $(\{\mathtt{p}\}, \{\{\mathtt{p}\}, \{\mathtt{p}, \mathtt{q}\}\}, \{\{\mathtt{q}\}, \{\mathtt{p}, \mathtt{q}\}\})$. In this structure,* p *is satisfied, but* q *is not,* $\mathbf{K}\,\mathtt{p}$ *is satisfied but* $\mathbf{K}\,\mathtt{q}$ *is not, and* $\mathbf{not}\,\mathtt{p}$ *is satisfied but* $\mathbf{not}\,\mathtt{q}$ *is not. Hence,* $\mathbf{K}\,\mathtt{p}$ *is known and not known at the same time.*

Such problems are resolved by considering only certain structures. In fact, MKNF interpretations, as defined next, are based on structures where the two sets, M and N are identical.

Definition 4.6. *An* MKNF interpretation M *over a universe* Δ *is a nonempty set of first-order interpretations. For a closed MKNF formula* φ*, we say that* M satisfies φ*, written* $M \models \varphi$*, if* $(I, M, M) \models \varphi$ *for each* $I \in M$*.*

In the literature, this notion is also called an *S5 model*, in which **not** can be replaced by $\neg\mathbf{K}$. The notion of a (non-monotonic) two-valued MKNF model of a closed MKNF formula φ is based on a preference relation on MKNF interpretations that satisfy φ.

Definition 4.7. *Let* φ *be a closed MKNF formula. An MKNF interpretation* M *over* Δ *is a* two-valued MKNF model *of* φ *if*

(1) M *satisfies* φ*, and*

(2) for each MKNF interpretation M' *such that* $M' \supset M$ *we have* $(I', M', M) \not\models \varphi$ *for some* $I' \in M'$*.*

An MKNF formula φ *is* MKNF satisfiable *if a two-valued MKNF model of* φ *exists; otherwise* φ *is* MKNF unsatisfiable*. Furthermore,* φ MKNF entails ψ*, written* $\varphi \models_{MKNF} \psi$*, if* $M \models \psi$ *for each two-valued MKNF model* M *of* φ*.*

Note that this definition of model is asymmetric in the treatment of the modal operators **K** and **not**. In fact, the maximization of M in (2) is only done in the component of the structure used for evaluating the operator **K**. This results in a minimization of the derivable **K**-atoms in any two-valued MKNF model of a given formula φ.

Example 4.8. *Though* $M = \{\mathtt{p}\}\}$ *satisfies both* $\mathbf{K}\,\mathtt{p}$ *and* $\neg\mathbf{not}\,\mathtt{p}$*,* M *is only a two-valued MKNF model of the first formula.* M *is not a two-valued MKNF model of the second one since* $(I', M', M) \models \neg\mathbf{not}\,\mathtt{p}$ *holds for any* M' *with* $M' \supset M$*. In fact,* $\neg\mathbf{not}\,\mathtt{p}$ *does not have an MKNF model.*

4.3 Standard Names Assumption

The MKNF semantics, as originally defined in [Lifschitz, 1991], shows certain undesirable properties such as counterintuitive semantics caused by the usage of arbitrary

universes and the differing interpretation of constants in different interpretations. To overcome these problems, [Motik and Rosati, 2010] additionally applies the standard name assumption to hybrid MKNF knowledge bases. We briefly recall two such problems from [Motik and Rosati, 2010] and the notion of standard names assumption introduced to overcome them. For the complete discussion of this issue, we refer to [Motik and Rosati, 2010].

One problem when using MKNF as in [Lifschitz, 1991] for the integration of rules and ontologies is the usage of arbitrary universes. Consider the MKNF formula $\varphi = \varphi_1 \wedge \varphi_2$, where $\varphi_1 = \mathbf{K}\,\mathtt{P}(\mathtt{a})$ and $\varphi_2 = \mathbf{not}\,\mathtt{P}(\mathtt{b}) \supset \mathbf{f}$. Intuitively, one would not expect that φ is satisfiable since there is no indication that $\mathtt{P}(\mathtt{b})$ should be true. However, if the universe contains only one element, then $\mathtt{a}$ and $\mathtt{b}$ are interpreted as the same object, and φ is satisfied. In this case one unintendedly derives that $\varphi \models \mathtt{a} \approx \mathtt{b}$ holds.

Another problem is caused by constants that are interpreted differently in different interpretations. Consider $\varphi_1 = \mathbf{K}\,\mathtt{P}(\mathtt{a})$ and $\varphi_2 = \exists \mathtt{x} : \mathbf{K}\,\mathtt{P}(\mathtt{x})$. In this case one would expect that $\varphi_1 \models \varphi_2$, that is, every two-valued MKNF model of φ_1 is also a two-valued MKNF model of φ_2. However, let M be an MKNF interpretation containing two elements I_1 and I_2 where I_1 is a first-order interpretation in which $\mathtt{a}$ is interpreted as a name α_1 and $I_1 \models \mathtt{P}(\alpha_1)$, and I_2 is a first-order interpretation in which $\mathtt{a}$ is interpreted as some other name α_2 and $I_2 \models \mathtt{P}(\alpha_2)$. We thus have that $M \models \varphi_1$ but not $M \models \varphi_2$ since this would require to have an $\mathtt{x}$ in the domain such that $\mathtt{P}(\mathtt{x})$ is true in all $I \in M$.

To avoid such unintended behavior, the standard name assumption is imposed on top of MKNF.

Definition 4.9. *(Standard Name Assumption [Motik and Rosati, 2010]). A first-order interpretation I over a signature Σ employs the* standard name assumption *if*

(1) the universe Δ of I contains all constants of Σ and a countably infinite number of additional constants called parameters*;*

(2) $t^I = t$ for each ground term t constructed using the function symbols from Σ and the constants from Δ; and

(3) the predicate $\approx$ is interpreted in I as a congruence relation – that is, $\approx$ is reflexive, symmetric, transitive, and allows for the replacement of equals by equals [Fitting, 1996].

Consequences of first-order formulas under the standard first-order semantics and the standard name assumption cannot be distinguished [Motik and Rosati, 2010]. Thus, we can consider first-order inferences to be based on the standard first-order semantics, even though technically the standard name assumption is applied.

4.4 Hybrid MKNF$^+$ Knowledge Bases

Hybrid MKNF knowledge bases, as introduced in [Motik and Rosati, 2010], essentially consist of two components: a first-order theory, such as a theory in description logics, and a finite set of rules of modal atoms (similar to rules of ASP) of the following form, where the symbol $\leftarrow$ corresponds to the operator $\subset$ in MKNF:

$$\mathbf{K}\, H_1 \vee \ldots \vee \mathbf{K}\, H_l \leftarrow \mathbf{K}\, A_1, \ldots, \mathbf{K}\, A_n, \mathbf{not}\, B_1, \ldots, \mathbf{not}\, B_m \tag{4.2}$$

However, such knowledge bases are not general enough to capture several related approaches combining ontologies and rules, so the more general MKNF$^+$ knowledge bases were defined in [Motik and Rosati, 2010] to allow for the usage of arbitrary first-order formulas in the modal atoms and a mixture of modal and first-order atoms in rules. Next, we are going to recall MKNF$^+$ knowledge bases.[1]

Definition 4.10. *A* generalized atom *is a first-order formula. A generalized atom is* ground *if it does not contain free variables. A generalized atom ξ_G is a* grounding *of a generalized atom ξ if ξ_G is obtained from ξ by replacing its free variables with constants. A* generalized atom base *GHB is a set of generalized atoms such that $\xi \in GHB$ implies $\xi_G \in GHB$ for each grounding ξ_G of ξ.*

An MKNF$^+$ atom *over GHB is either a* nonmodal atom *of the form ξ, a* $\mathbf{K}$*-atom of the form $\mathbf{K}\,\xi$ or a* **not***-atom of the form* **not** *ξ, where $\xi \in GHB$. An* MKNF$^+$ rule *r over GHB is a formula of the following form, where each H_k and each A_i is a nonmodal or a* $\mathbf{K}$*-atom over GHB and each* **not** *B_j a* **not***-atom over GHB.*

$$H_1 \vee \cdots \vee H_l \leftarrow A_1, \ldots, A_n, \mathbf{not}\, B_1, \ldots, \mathbf{not}\, B_m \tag{4.3}$$

The set $\mathcal{H} = \{\mathbf{K}\, H_k \mid 1 \leq k \leq l\}$ is called the head *of r, and the set $\mathcal{B} = \{A_j \mid 1 \leq i \leq n\} \cup \{\mathbf{not}\, B_j \mid 1 \leq j \leq m\}$ is called the* body *of r. We may split the body into its positive part $\mathcal{B}^+ = \{A_j \mid 1 \leq i \leq n\}$ and its negative part $\mathcal{B}^- = \{\mathbf{not}\, B_j \mid 1 \leq j \leq m\}$*

[1] All definitions in the rest of the thesis are parameterized by a signature containing the equality predicate $\approx$.

and abbreviate rules by $\mathcal{H} \leftarrow B$ *or* $\mathcal{H} \leftarrow B^+ \wedge B^-$ *An MKNF*$^+$ *rule* r *is* non-disjunctive *if* $l = 1$, r *is* positive *if* $m = 0$, *and* r *is a* fact *if* $n = m = 0$. *If* $l = 0$, *then we write the head as* $\mathbf{f}$ *and such a rule is called an* integrity constraint. *Moreover, an MKNF*$^+$ *rule is* safe *if each variable that occurs free in some atom also occurs free in a body* $\mathbf{K}$*-atom. A* program $\mathcal{P}$ *is a finite set of MKNF*$^+$ *rules.*

Let $\mathcal{DL}$ *be a description logic and* GHB *a generalized atom base. A* hybrid MKNF$^+$ knowledge base *over* $\mathcal{DL}$ *and* GHB *is a pair* $\mathcal{K}$ *is a pair* $(\mathcal{O}, \mathcal{P})$ *where* $\mathcal{O} \in \mathcal{DL}$ *is a DL knowledge base and* $\mathcal{P}$ *is a program over* GHB.

From now onwards, we assume that all hybrid MKNF$^+$ knowledge bases are parameterized by a description logic $\mathcal{DL}$ and a generalized atom base GHB.

Hybrid MKNF$^+$ knowledge bases, as defined above, do not coincide syntactically with any MKNF formula. For interpreting hybrid MKNF knowledge bases in terms of MKNF logic, the transformation π that transforms a DL ontology into first-order formulas[1] is extended to knowledge bases as follows:

Definition 4.11. *Let* $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ *be a hybrid MKNF*$^+$ *knowledge base and* $\pi(\mathcal{O})$ *be the transformation of* $\mathcal{O}$ *into a formula of first-order logic with equality. We extend* π *to rules* r *of the form (4.3),* $\mathcal{P}$, *and* $\mathcal{K}$ *as follows, where* $\vec{x}$ *is the vector of the free variables of* r.

$$\pi(r) = \forall \vec{x} : (A_1 \wedge \ldots \wedge A_n \wedge \mathbf{not}\, B_1 \wedge \ldots \wedge \mathbf{not}\, B_m \supset H_1 \vee \ldots \vee H_l)$$

$$\pi(\mathcal{P}) = \bigwedge_{r \in \mathcal{P}} \pi(r) \qquad \pi(\mathcal{K}) = \mathbf{K}\, \pi(\mathcal{O}) \wedge \pi(\mathcal{P})$$

Note that the A_i and H_k may be $\mathbf{K}$-atoms. To simplify the presentation, we abuse notation and, when this does not cause confusion, identify $\mathcal{K}$ with $\pi(\mathcal{K})$. It shall be obvious from the context when $\mathcal{K}$ represents its first-order transformation $\pi(\mathcal{K})$.

4.5 Semantic Properties of Hybrid MKNF$^+$

Obviously, the formalism just recalled immediately satisfies tightness and flexibility, i.e., two of the criteria for a combination of rules and ontologies we want to be satisfied. We discuss decidability separately in the next section since the approach as such, without any restrictions, is undecidable. First, we focus our attention on faithfulness.

[1] See, e.g., [Baader et al., 2007a] for such a transformation.

As presented in [Motik and Rosati, 2010], if we consider that $\mathcal{K} = (\mathcal{O}, \emptyset)$, i.e., the rule part is empty, then it is easy to see that $\mathcal{K} \models_{MKNF} \psi$ if and only if $\mathcal{O} \models \psi$. The reason is that logical consequences under $\mathcal{O}$ and $\mathbf{K}\,\mathcal{O}$ are the same because $\mathcal{O}$ itself does not contain any modal operators.

For the other case, where $\mathcal{K} = (\emptyset, \mathcal{P})$, i.e., the ontology is empty, we can rely on the results [Lifschitz, 1991, 1994], which can even be simplified since we assume the standard name assumption [Motik and Rosati, 2010]. A rule r in ASP of the form 2.2 with a vector of the free variables $\vec{x}$ is straightforwardly transformed into an MKNF formula $\pi(r)$ as follows.

$$\forall \vec{x} : (\mathbf{K}\,A_1 \wedge \ldots \wedge \mathbf{K}\,A_n \wedge \mathbf{not}\,B_1 \wedge \ldots \wedge \mathbf{not}\,B_m \supset \mathbf{K}\,H_1 \vee \ldots \vee \mathbf{K}\,H_l) \tag{4.4}$$

This transformation can be extended to an ASP program in very much the same way as in Definition 4.11 and, thus, faithfulness holds.

Of course, strictly speaking, this correspondence does not hold for any arbitrary rule part, since a program in MKNF$^+$ is more general than ASP programs. However, it was shown in [Motik and Rosati, 2010] that nonmodal atoms can be removed from hybrid MKNF$^+$ knowledge bases resulting in the notion of hybrid MKNF knowledge bases.

Definition 4.12. MKNF rules *and* MKNF knowledge bases *are defined as in Definition 4.10 with the difference that each atom* H_k, $1 \leq k \leq l$, *and each atom* A_i, $1 \leq i \leq n$, *in each rule must be a modal* **K***-atom.*

It was shown in [Motik and Rosati, 2010] that there is a transformation such that the original hybrid MKNF$^+$ knowledge base $\mathcal{K}^+$ has exactly the same two-valued MKNF models as the resulting hybrid MKNF knowledge base. We do not present the full proof here, but rather sketch the idea. Essentially we can rewrite MKNF$^+$ rules such that the nonmodal and the modal **K**-atoms are separated.

$$\bigvee H_{k1} \vee \bigvee \mathbf{K}\,H_{k2} \leftarrow \bigwedge A_{i1} \wedge \bigwedge \mathbf{K}\,A_{i2} \wedge \bigwedge \mathbf{not}\,B_j \tag{4.5}$$

Given $\pi(r)$ for such a rule r, we can obtain the transformed formula, where $\vec{y}$ is the list of variables that occur only in H_{k1} and A_{i1}, and $\vec{x}$ is the list of all other variables in r.

$$\forall \vec{x} : \left(\bigwedge \mathbf{K}\,A_{i2} \wedge \bigwedge \mathbf{not}\,B_j \supset \mathbf{K}\,(\forall \vec{y} : \bigvee H_{k1} \vee \bigvee \neg A_{i1}) \vee \bigvee \mathbf{K}\,H_{k2}\right) \tag{4.6}$$

Thus, nonmodal atoms can be encapsulated in one **K**-atom allowing us to separate first-order and non-monotonic reasoning. Therefore, we consider only hybrid MKNF knowledge bases.

We now recall two examples to show semantic differences between first-order and MKNF logics and we refer for the details to [Motik and Rosati, 2010]. First, let us consider an MKNF formula $(\exists \mathtt{x} : \mathtt{A}(\mathtt{x})) \wedge (\mathbf{K}\,\mathtt{A}(\mathtt{x}) \supset \mathbf{K}\,\mathtt{p})$. We may expect that $\mathbf{K}\,\mathtt{p}$ holds. However, $\mathtt{x}$ may vary in each interpretation while $\mathbf{K}\,\mathtt{A}(\mathtt{x})$ asks whether we know one $\mathtt{x}$ such that $\mathbf{K}\,\mathtt{A}(\mathtt{x})$ holds. Thus, in MKNF $\mathbf{K}\,\mathtt{p}$ does not hold. This is a general issue and one consequence for hybrid MKNF is that MKNF rules can essentially only be used for derivations in case of names that are present in the knowledge base. The ability to derive terminological consequences is also affected since, e.g., $\mathbf{K}\,\mathtt{B}(\mathtt{x}) \leftarrow \mathbf{K}\,\mathtt{A}(\mathtt{x})$ does not imply $\forall \mathtt{x} : (\mathtt{A}(\mathtt{x}) \supset \mathtt{B}(\mathtt{x}))$. We note that this ability is not completely absent, and given an appropriately expressive DL, certain conclusions can still be drawn.

We finish this section by pointing out that the closely related MBNF logics [Lifschitz, 1994] is not considered for combining rules and ontologies since in the formal definitions of the semantics the sets I and I' do not have to appear in M and M' (see Definition 4.7). As a consequence, $\mathbf{K}\,\pi(\mathcal{O})$ and nonmodal atoms would not interact so that first-order rule extensions of DL would not be captured, and φ and $\mathbf{K}\,\varphi$ would not be equivalent any longer [Motik and Rosati, 2010].

4.6 Decidability for Hybrid MKNF

Hybrid MKNF is in general undecidable, so we recall the restrictions that are necessary to ensure that a decidable formalism is obtained.

First, the DL has to be decidable. More precisely, the approach of hybrid MKNF knowledge bases is applicable to any first-order fragment $\mathcal{DL}$ satisfying the subsequent list of conditions:

(i) each knowledge base $\mathcal{O} \in \mathcal{DL}$ can be translated into a formula $\pi(\mathcal{O})$ of function-free first-order logic with equality;

(ii) $\mathcal{DL}$ supports *A-Box*-assertions of the form $P(a_1, \ldots, a_n)$, where P is a predicate and each a_i a constant of $\mathcal{DL}$;

(iii) satisfiability checking and instance checking, i.e., checking entailments of the form $\mathcal{O} \models P(a_1, \ldots, a_n)$, are decidable.

The limitation to function-free first-order logic is necessary to ensure decidability. Of course, this applies equally to rules. So, for the rest of the thesis, we only consider DLs that satisfy the conditions above, and we do not allow function symbols in hybrid MKNF knowledge bases.

Unfortunately, hybrid MKNF knowledge bases, even with these assumptions, are in general undecidable, unless they are restricted in some way. In [Motik and Rosati, 2010] two reasons for that were identified: the appearance of complex first-order formulas, such as conjunctive queries in the head of the rules; and default negation that allows the application of rules to the entire domain even if these are safe.

We now recall the notions that are defined in [Motik and Rosati, 2010] to overcome these problems. In the case of the second problem, the usual solution is to restrict the applicability of rules to those individuals/constants appearing explicitly in the knowledge base. This restriction is achieved by DL-safety.

Definition 4.13. *Let $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ be a hybrid MKNF knowledge base and the signature Σ contain a subset $\Sigma_{DL} \subseteq \Sigma$ such that $\approx \in \Sigma_{DL}$. The predicates in Σ_{DL} are called* DL-predicates *and they have to appear in $\mathcal{O}$. The predicates in $\Sigma \setminus \Sigma_{DL}$ are called* non-DL-predicates *and do only appear in $\mathcal{P}$.*

A generalized atom ξ is a DL-atom *if it contains only predicates of Σ_{DL}, and ξ is a* non-DL-atom *if it is of the form $(\neg)P(t_1, \ldots, t_n)$ where P is a non-DL-predicate.*[1]

An MKNF rule r is DL-safe *if each modal atom in it is of the form $\mathbf{K}\,A$ or $\mathbf{not}\,A$, where A is either DL-atom or a non-DL-atom, and if every variable in r occurs in at least one non-DL-atom $\mathbf{K}\,B$ in the body of r. We say that $\mathcal{K}$ is* DL-safe *if all the rules in $\mathcal{K}$ are DL-safe.*

In practice, DL-safety can be achieved by introducing a new (special) non-DL-predicate, say O, adding an assertion $\mathbf{K}\,O(a)$ for each individual a that appears in the knowledge base, and adding to each rule r a modal atom $\mathbf{K}\,O(x)$ for each variable x appearing in r. This transformation and its effects have been discussed in [Motik et al., 2005], and, as argued in [Motik and Rosati, 2010], it does not affect the semantics of MKNF rules.

[1] Note that there are atoms that are neither DL-nor non-DL-atoms.

We can thus ground the knowledge base, and we only consider ground knowledge bases for the definition of the semantics.

Definition 4.14. *Let $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ be a hybrid MKNF knowledge base. The* ground instantiation *of $\mathcal{K}$ is the KB $\mathcal{K}_G = (\mathcal{O}, \mathcal{P}_G)$ where $\mathcal{P}_G$ is obtained from $\mathcal{P}$ by replacing each rule r of $\mathcal{P}$ with a set of rules substituting each variable in r with constants from $\mathcal{K}$ in all possible ways.*

It was shown in [Motik and Rosati, 2010] that, for a DL-safe hybrid knowledge base $\mathcal{K}$, the two-valued MKNF models of $\mathcal{K}$ and $\mathcal{K}_G$ coincide.

The combined conditions for decidability are recalled in Definition 4.15.

Definition 4.15. *Let $\mathcal{DL}$ be a description logic, GHB a generalized atom base, and $\mathcal{H}$ a subset of GHB. Then, $\mathcal{DL}$, GHB, and $\mathcal{H}$ are* admissible *if, for each $\mathcal{O} \in \mathcal{DL}$, each finite set $S \subseteq \mathcal{H}$ of ground generalized atoms, each finite set N of assertions of the form $a \not\approx b$, each each generalized atom $\xi \in GHB$, checking whether $\mathcal{O} \cup S \cup N \models \xi$ is decidable.*

An MKNF knowledge $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ over $\mathcal{DL}$ and GHB is admissible *if $\mathcal{DL}$, GHB, and $\mathcal{H}$ are admissible, $\mathcal{K}$ is DL-safe and finite, and each rule in $\mathcal{P}$ contains only generalized atoms from $\mathcal{H}$ in its head.*

This complex condition fixes the remaining problem by restricting first-order reasoning in the DL together with the first-order formulas possibly appearing in the heads of the rules to be decidable.

In [Motik and Rosati, 2010], several reasoning algorithms were provided for combinations of arbitrary description logic fragments and rules of differing expressivity, and their computational complexity was studied. As argued in [Motik and Rosati, 2010], given that rules are only applicable to known individuals, the principal application of hybrid MKNF is answering instance queries over individuals. In such cases, it is expected that the size of the data is much larger than the size of the conceptual knowledge. Thus, the data complexity provides a much better estimate of the complexity.

Fig. 4.1 presents the data complexity of instance checking for several combinations of rules (with or without disjunction, and with arbitrary or stratified[1] negation or without **not** in the rules) with description logics fragments of differing computational

[1]Essentially, rules can be separated into strata that can be evaluated separately – see Section 4.4 in [Motik and Rosati, 2010].

$\vee$	**not**	no $\mathcal{DL}$	$\mathcal{C}_{\mathcal{DL},\mathcal{B},\mathcal{H}} = \text{PTIME}$	$\mathcal{C}_{\mathcal{DL},\mathcal{B},\mathcal{H}} = \text{coNP}$
no	no	PTIME	PTIME	coNP/coDP
no	stratified	PTIME	PTIME	Δ_2^p
no	yes	coNP	coNP	Π_2^p
yes	no	coNP/Π_2^p	coNP/Π_2^p	coNP/Π_2^p
yes	yes	Π_2^p	Π_2^p	Π_2^p

Figure 4.1: Data complexity of instance checking in Admissible MKNF KBs.
The first two columns determine the expressiveness of the rules, and columns three to five show the data complexity of instance checking when the DL is absent, and when it is complete for PTIME and coNP, respectively. Entries with two results differ on entailment of $\mathbf{K}\,A$ and $\neg\mathbf{K}\,A$.

complexity. Note that this complexity of the DL is affected by the notion of admissibility (Definition 4.15).

We point out that if we disallow disjunction, then allowing arbitrary non-monotonic negation increases the data complexity drastically and in particular beyond tractability. As already mentioned, this is precisely what we improve in this thesis: non-stratified rules do not constitute a problem regarding complexity resulting in an approach that is robust for knowledge base engineering and for evolution of knowledge bases.

4.7 Hybrid MKNF with Normal Rules

Hybrid MKNF knowledge bases (and also the even more general hybrid MKNF$^+$ knowledge bases) are defined in a quite general way, also to capture a large variety of related approaches. It was pointed out in [Motik and Rosati, 2010] that for practical purposes, generalized atoms should include first-order atoms in rules (as in MKNF$^+$ KBs), classical negation (to capture ASP) and conjunctive queries.

Even assuming that a concrete combination of a certain DL with some kind of rules is admissible, the availability of certain constructs in MKNF rules may raise the computational complexity due to the tight integration of the two components. For example, in the polynomial DL $\mathcal{SROEL}$, classical negation is not allowed, so if the operator $\neg$ appears in the rules, then the first-order-reasoning component is no longer polynomial. As another example, conjunctive query answering is usually of a higher

computational complexity than standard reasoning tasks, such as instance checking.

Since our objective is to obtain a formalism that is highly efficient, we concentrate here on a fragment of rules that corresponds to normal rules in Logic Programming. In other words, we do not consider modal atoms that do contain anything but a simple first-order atom. Of course, certain constructors, such as $\neg$, may be permitted in case the DL in consideration allows their usage. In this case, given some modal atom $\mathbf{K}\,\neg P$ we can simply introduce a new predicate Q and substitute $\mathbf{K}\,\neg P$ by $\mathbf{K}\,Q$ and add $Q \equiv \neg P$ to the ontology (see [Motik and Rosati, 2010]).

Another problem on our quest for an efficient formalism based on Well-Founded Semantics is disjunction in the head of the rules. The well-founded semantics for logic programs, originally defined in [van Gelder et al., 1991], only applies to non-disjunctive logic programs, and there is no established WFS that allows for disjunction in the rule heads (see, e.g., [Knorr and Hitzler, 2007]). Moreover, disjunctions, such as $\mathbf{K}\,H_1 \vee \cdots \vee \mathbf{K}\,H_l$, in the rule head enforce that one of the $\mathbf{K}\,H_k$ is true in interpretations of an MKNF model. In other words, such disjunctions result in various models, which is conceptually orthogonal to the idea of WFS that relies on the computation of one unique model. The proposals for well-founded semantics of logic programs including disjunctions try to overcome that by applying reasoning techniques that may be considered first-order to some extend, but, as already pointed out, no convincing solution has been presented to the problem. To avoid this problem, and to be compatible with the well-founded semantics of normal programs, we restrict to rules that are non-disjunctive.

Therefore, in the rest of the thesis, which contains our main results, we assume that all hybrid MKNF knowledge bases contain only non-disjunctive rules, i.e., no disjunction occurs in the head of any rule, and that modal atoms do only contain first-order atoms. Now, we present this simplified notion of hybrid MKNF knowledge bases (and a few related ones) to ease the reading.

Definition 4.16. *Let $\mathcal{O}$ be a DL knowledge base. A function-free first-order atom $P(t_1, \ldots, t_n)$ over Σ such that P is $\approx$ or occurs in $\mathcal{O}$ is called a DL-atom; all other atoms are called non-DL-atoms. An* MKNF rule *r has the following form where H, A_i, and B_i are function free first-order atoms:*

$$\mathbf{K}\,H \leftarrow \mathbf{K}\,A_1, \ldots, \mathbf{K}\,A_n, \mathbf{not}\,B_1, \ldots, \mathbf{not}\,B_m \tag{4.7}$$

A program is a finite set of MKNF rules, and a hybrid MKNF knowledge base *$\mathcal{K}$ is a pair $(\mathcal{O}, \mathcal{P})$.*

We finish this section and the chapter by presenting a hybrid MKNF knowledge base abiding this restrictions in the setting of our example scenario.

Example 4.17. *Consider again the scenario of Example 1.1, together with the axioms and assertions of Example 3.18. These axioms can be part of an ontology $\mathcal{O}$ of a hybrid MKNF knowledge base $\mathcal{K} = (\mathcal{O}, \mathcal{P})$. In $\mathcal{P}$ we can encode further recommendation guidelines, in particular those that require closed world reasoning. For example, imagine that we want to give customers recommendations for interesting CDs they do not own and that do not have a low evaluation. This can be encoded with the rules shown below:*[1]

$$\begin{aligned} \mathbf{K}\,\mathtt{Recommend(x)} \leftarrow\ & \mathbf{K}\,\mathtt{CD(x)}, \mathbf{not}\,\mathtt{owns(x)}, \mathbf{not}\,\mathtt{LowEval(x)}, \\ & \mathbf{K}\,\mathtt{interesting(x)}. \end{aligned} \tag{4.8}$$

$$\begin{aligned} \mathbf{K}\,\mathtt{interesting(x)} \leftarrow\ & \mathbf{K}\,\mathtt{CD(x)}, \mathbf{K}\,\mathtt{CD(y)}, \mathbf{K}\,\mathtt{owns(y)}, \mathbf{not}\,\mathtt{owns(x)}, \\ & \mathbf{K}\,\mathtt{similar(x,y)}. \end{aligned} \tag{4.9}$$

$$\begin{aligned} \mathbf{K}\,\mathtt{similar(x,y)} \leftarrow\ & \mathbf{K}\,\mathtt{CD(x)}, \mathbf{K}\,\mathtt{CD(y)}, \mathbf{K}\,\mathtt{Artist(z)}, \\ & \mathbf{K}\,\mathtt{HasArtist(x,z)}, \mathbf{K}\,\mathtt{HasArtist(y,z)}. \end{aligned} \tag{4.10}$$

$$\mathbf{K}\,\mathtt{owns(EnConcert)} \leftarrow \tag{4.11}$$

$$\mathbf{K}\,\mathtt{HasArtist(EnConcert, JackJohnson)} \leftarrow \tag{4.12}$$

$$\mathbf{K}\,\mathtt{HasArtist(ToTheSea, JackJohnson)} \leftarrow \tag{4.13}$$

$$\mathbf{K}\,\mathtt{OnOffer(BNAW)} \leftarrow \tag{4.14}$$

Note that closed world reasoning is used for `owns` *and* `lowEval`*. In the case of predicate* `owns`*, it is reasonable to assume that the knowledge about owned CDs is fully available. So, if there is no fact stating that a given CD is owned, one should assume that the CD is not owned. In the case of predicate* `lowEval` *it might happen that there is no evaluation yet available, and we want the recommendation anyway: a CD is not considered for recommendation only when there actually is a (known) low evaluation for the CD. Such an evaluation could be taken from other customers of the store or from a web page of professional reviews. Here, for simplicity, we keep this part of the reasoning process implicit.*

[1] In this encoding, for simplicity, we consider that the program part of the knowledge base is specific to each customer. This avoids an explicit representation of several customers, of the relation stating which CDs are owned by each customer, etc.

Moreover, in the rules above, a CD is interesting if the customer owns another CD which is similar (4.9), and two CDs are similar if they have a common artist (4.10). Note that the predicate `CD` *is used to ensure DL-safety, and we assume that the instances of that predicate relevant to any drawn conclusion are always appropriately defined.*

If we now add facts (4.11)–(4.14), then we can derive `Recommend(ToTheSea)`*, since no low evaluation is known for* `ToTheSea`*, and* `Recommend(BNAW)` *since* `BNAW` *is on offer.*

This example illustrates that hybrid MKNF knowledge bases allow us to obtain consequences for predicates that are 'defined' both in the ontology and in the rules. The result may then be further applied to derive subsequent consequences either in rules or in the ontology. Note that the facts (4.12) and (4.13) are here explicitly added, representing the implicit consequences derivable from the appropriate ontology alone, similar to `HasArtist(BNAW, JohnColtrane)` *in Example 3.18.*

The rules in Example 4.17 are stratified. However, this example contains just an initial set of rules for our running example, which is further elaborated in the following chapters and becomes non-stratified, e.g., the addition of rule 6.11 in Example 6.16 renders the set of rules non-stratified.

Part II

A Well-founded Semantics for Hybrid MKNF Knowledge Bases

5

A Three-valued MKNF Semantics

Hybrid MKNF knowledge bases rely on the MKNF logics defined in [Lifschitz, 1991]. However, MKNF logic is a two-valued logics, and we claim that such a two-valued logic is not sufficient to provide the necessary means for a semantics that is related to the well-founded semantics for logic programs.

Therefore, we introduce a three-valued semantics for hybrid MKNF knowledge bases. The rationale and the main goal behind this three-valued semantics is to define a semantics that is closely related to the well-founded semantics of logic programs, but that, at the same time, extends the original (two-valued) MKNF semantics [Lifschitz, 1991] as presented in the previous chapter. This is done in order to take advantage of the (data) complexity of the WFS that is lower than the (data) complexity of the corresponding two-valued semantics in Logic Programming. Nevertheless, the DL-part of a hybrid MKNF knowledge base is still interpreted under the two-valued first-order semantics. Thus, we achieve a faithful integration, in the sense that without rules the meaning of the knowledge base exactly coincides with the usual semantics from DLs, and without DL we are able to establish a relation to the WFS (see Chapter 6).

We want to point out that the definition of the three-valued semantics presented in this chapter applies equally with or without the standard name assumption. However, since we want to achieve a semantics that is faithful w.r.t. the two-valued MKNF semantics as presented in Chapter 4, and that avoids counterintuitive results as described there, we assume standard name assumption.

We start this chapter by extending the notion of MKNF structures to three truth values and by defining evaluation in such structures (Section 5.1). Then, we extend the notion of MKNF interpretation and provide an appropriate model notion (Section 5.2). We show some general properties of our newly defined semantics (Section 5.3), and we introduce the notion of one particular three-valued MKNF model, namely the well-founded MKNF model (Section 5.4). We finish with a detailed discussion on the relation to the two-valued MKNF semantics (Section 5.5).

5.1 Evaluation in MKNF Structures

The two-valued hybrid MKNF semantics [Motik and Rosati, 2010] is closely related (cf. [Lifschitz, 1991]) to the stable models semantics [Gelfond and Lifschitz, 1988], the answer set semantics [Gelfond and Lifschitz, 1991] respectively, as seen in the previous chapter. In both of them, the meaning of a knowledge base is determined by a set of models. In fact, MKNF formulas, such as $\varphi = ((\mathbf{not}\,\mathtt{p} \supset \mathbf{K}\,\mathtt{q}) \wedge (\mathbf{not}\,\mathtt{q} \supset \mathbf{K}\,\mathtt{p}))$ (and the corresponding set of MKNF rules), have two models – one model in which $\mathtt{p}$ is true and $\mathtt{q}$ is false, and another one in which $\mathtt{p}$ is false and $\mathtt{q}$ is true. Moreover, these two-valued models are, in general, obtained by a guess and check process, thus having, in general, a computational complexity of at least NP.

The well-founded semantics of logic programs [van Gelder et al., 1991] generalizes the two-valued models of the stable model semantics to a three-valued setting. In this way, it is possible to determine the meaning of a logic program solely on the basis of a single (minimal) model that is obtained with a lower computational complexity. Intuitively, a third truth value $\mathbf{u}$, denoting undefined, is introduced as an alternative to the values $\mathbf{t}$ and $\mathbf{f}$, enabling one to delay the evaluation to any of the two latter values until further information is available. We want to follow this idea when defining a three-valued MKNF semantics. There is however one more problem to be taken into account: since we are interested in applying the semantics to hybrid MKNF knowledge bases containing two-valued ontologies, which we want to integrate faithfully, we are going to define the semantics in such a way that an MKNF formula corresponding to a DL fragment is ensured to be just two-valued.

We therefore define a three-valued MKNF semantics that extends the two-valued semantics of [Motik and Rosati, 2010], but remains two-valued for the case of MKNF

formulas without modal operators. We start by defining MKNF structures for this three-valued setting.

Definition 5.1. *A* three-valued MKNF structure $(I, \mathcal{M}, \mathcal{N})$ *consists of a first-order interpretation* I *and two pairs* $\mathcal{M} = \langle M, M_1 \rangle$ *and* $\mathcal{N} = \langle N, N_1 \rangle$ *of sets of first-order interpretations where* $M_1 \subseteq M$ *and* $N_1 \subseteq N$*. An MKNF structure is called* total *if* $\mathcal{M} = \langle M, M \rangle$ *and* $\mathcal{N} = \langle N, N \rangle$*.*

In the two-valued semantics, an MKNF structure (I, M, N) contains sets of interpretations M and N for evaluating a modal atom $\mathbf{K}\,\varphi$, respectively $\mathbf{not}\,\varphi$, to $\mathbf{t}$ or $\mathbf{f}$, depending on whether φ is contained in all elements of M, respectively N. This clearly leaves no space for an extension to a third truth value $\mathbf{u}$. So, we turn sets of interpretations into pairs of sets of interpretations. Then, as we show below, a modal atom $\mathbf{K}\,\varphi$ is true w.r.t. $\langle M, M_1 \rangle$ if φ is true in all elements of M. Alternatively, a modal atom $\mathbf{K}\,\varphi$ is false if φ is not true in all elements of M_1, and such a modal atom $\mathbf{K}\,\varphi$ is undefined otherwise, i.e, if φ is true in all elements of M_1 but not in all elements of M. The additional restrictions, saying that $M_1 \subseteq M$ and $N_1 \subseteq N$, are needed to ensure that no modal atom can be both true and false at the same time, and it can easily be shown via induction that the same holds for any MKNF formula φ. In this way, we guarantee that no fourth truth value 'both' is needed. Nevertheless, given an MKNF formula φ, three-valued MKNF structures may evaluate $\mathbf{K}\,\varphi$ and $\mathbf{not}\,\varphi$ to true at the same time, just like in the two-valued case, and we show below how to prevent this from happening when defining MKNF interpretation pairs.

We now define the evaluation of closed MKNF formulas in such three-valued MKNF structures.

Definition 5.2. *Let* $(I, \mathcal{M}, \mathcal{N})$ *be a three-valued MKNF structure and* $\{\mathbf{t}, \mathbf{u}, \mathbf{f}\}$ *the set of truth values with the order* $\mathbf{f} < \mathbf{u} < \mathbf{t}$*, where the operator* max *(resp.* min*) chooses the greatest (resp. least) element with respect to this ordering. We define:*

- $(I, \mathcal{M}, \mathcal{N})(P(t_1, \ldots, t_n)) = \begin{cases} \mathbf{t} & \text{iff } (t_1^I, \ldots, t_n^I) \in P^I \\ \mathbf{f} & \text{iff } (t_1^I, \ldots, t_n^I) \notin P^I \end{cases}$
- $(I, \mathcal{M}, \mathcal{N})(\neg\varphi) = \begin{cases} \mathbf{t} & \text{iff } (I, \mathcal{M}, \mathcal{N})(\varphi) = \mathbf{f} \\ \mathbf{u} & \text{iff } (I, \mathcal{M}, \mathcal{N})(\varphi) = \mathbf{u} \\ \mathbf{f} & \text{iff } (I, \mathcal{M}, \mathcal{N})(\varphi) = \mathbf{t} \end{cases}$

- $(I,\mathcal{M},\mathcal{N})(\varphi_1 \wedge \varphi_2) = \mathsf{min}\{(I,\mathcal{M},\mathcal{N})(\varphi_1),(I,\mathcal{M},\mathcal{N})(\varphi_2)\}$
- $(I,\mathcal{M},\mathcal{N})(\varphi_1 \supset \varphi_2) = \mathbf{t}$ *iff* $(I,\mathcal{M},\mathcal{N})(\varphi_2) \geq (I,\mathcal{M},\mathcal{N})(\varphi_1)$ *and* $\mathbf{f}$ *otherwise*
- $(I,\mathcal{M},\mathcal{N})(\exists x:\varphi) = \mathsf{max}\{(I,\mathcal{M},\mathcal{N})(\varphi[\alpha/x]) \mid \alpha \in \Delta\}$
- $(I,\mathcal{M},\mathcal{N})(\mathbf{K}\,\varphi) = \begin{cases} \mathbf{t} & \textit{iff } (J,\langle M,M_1\rangle,\mathcal{N})(\varphi) = \mathbf{t} \textit{ for all } J \in M \\ \mathbf{f} & \textit{iff } (J,\langle M,M_1\rangle,\mathcal{N})(\varphi) = \mathbf{f} \textit{ for some } J \in M_1 \\ \mathbf{u} & \textit{otherwise} \end{cases}$
- $(I,\mathcal{M},\mathcal{N})(\mathbf{not}\,\varphi) = \begin{cases} \mathbf{t} & \textit{iff } (J,\mathcal{M},\langle N,N_1\rangle)(\varphi) = \mathbf{f} \textit{ for some } J \in N_1 \\ \mathbf{f} & \textit{iff } (J,\mathcal{M},\langle N,N_1\rangle)(\varphi) = \mathbf{t} \textit{ for all } J \in N \\ \mathbf{u} & \textit{otherwise} \end{cases}$

As intended, this evaluation is not a purely three-valued one, since first-order atoms are evaluated as in the two-valued case. In fact, an MKNF formula φ without modal operators (and thus also a pure description logic knowledge base) is only two-valued. It can easily be seen that such a φ is evaluated in exactly the same way as in the two-valued case (Definition 4.4). This is desired in particular when the knowledge base consists just of the DL part. So, the third truth value only affects MKNF formulas containing modal atoms, which in the case of hybrid MKNF knowledge bases can only occur in the MKNF rules. These rules, corresponding to implications, are, however, no longer interpreted in a way one would expect from a boolean perspective: $\mathbf{u} \supset \mathbf{u}$ is true in the evaluation defined above, while $\mathbf{u} \vee \neg\mathbf{u}$ is actually undefined. The reason for this change is that, in this way rules can only be true or false, similarly to what happens in Logic Programming, even if they contain undefined modal atoms. Intuitively, the advantage for hybrid MKNF knowledge bases is that we can leave single modal atoms undefined, thus not necessarily having to create several models, while the entire knowledge base is only true or false.

We point out that the evaluation of **not** w.r.t. $\langle N,N_1\rangle$ is symmetrical to the evaluation of **K** w.r.t. $\langle M,M_1\rangle$, only that the conditions are switched. E.g., the condition for true modal **K**-atoms w.r.t. M yields false modal **not**-atoms w.r.t. N. In case of $M = N$ and $M_1 = N_1$ this corresponds to the two-valued (monotonic) evaluation in Definition 4.4.

Example 5.3. *Recall Example 4.5, and consider the three-valued MKNF structure* $(\{\mathtt{p}\}, < \{\{\mathtt{p}\},\{\mathtt{p},\mathtt{q}\}\},\{\{\mathtt{p},\mathtt{q}\}\} >, < \{\emptyset,\{\mathtt{q}\},\{\mathtt{p},\mathtt{q}\}\},\{\{\mathtt{q}\},\{\mathtt{p},\mathtt{q}\}\} >)$*. Here, we have that* $\mathtt{p}$ *is true and* $\mathtt{q}$ *is false, just as in the two-valued case of Example 4.5. We also see that* $\mathbf{K}\,\mathtt{p}$ *and* $\mathbf{not}\,\mathtt{p}$ *are true, but here we obtain that both* $\mathbf{K}\,\mathtt{q}$ *and* $\mathbf{not}\,\mathtt{q}$ *are undefined.*

5.2 Three-valued MKNF Models

Given the definition of evaluation of MKNF formulas, we are now able to extend (two-valued) MKNF interpretations and MKNF models to three truth values. For that purpose, we have to generalize (two-valued) MKNF interpretations M to pairs of MKNF interpretations (M, N), since otherwise no formula could ever be undefined.

Definition 5.4. *An* MKNF interpretation pair (M, N) *consists of two MKNF interpretations* M*,* N *with* $\emptyset \subset N \subseteq M$*. An MKNF interpretation pair* satisfies *a closed MKNF formula* φ*, written* $(M, N) \models \varphi$*, if and only if*

$$(I, \langle M, N\rangle, \langle M, N\rangle)(\varphi) = \mathbf{t}$$

for each $I \in M$*. If* $M = N$*, then the MKNF interpretation pair* (M, N) *is called* total*. If there exists an MKNF interpretation pair satisfying* φ*, then* φ *is* consistent*.*

The intuition is that the set M contains all interpretations that model only truth, while N models everything that is true or undefined. Evidently, just as in the two-valued case, anything not being modeled in N is false. The subset relation between M and N ensures that MKNF interpretation pairs are defined in accordance with the three-valued MKNF structures, so that each formula is evaluated to exactly one truth value. Note the striking similarity compared to (two-valued) MKNF interpretations in the two-valued case by using the MKNF interpretation pair (M, N) to evaluate both **K** and **not** simultaneously. This similarity becomes even more apparent if we consider a total interpretation pair (M, M) and we show in Section 5.5 that a correspondence to evaluation in the two-valued case holds.

Example 5.5. *Consider the formula* $\varphi = (\mathtt{p} \vee \mathtt{q}) \wedge \mathbf{not}\,\mathtt{q}$ *that requires that* $\mathtt{q}$ *must not be known. Obviously,* $M_1 = (\{\{\mathtt{p},\mathtt{q}\},\{\mathtt{q}\}\},\{\{\mathtt{q}\}\})$ *does not satisfy* φ *for that reason. However,* $M_2 = (\{\emptyset,\{\mathtt{p}\}\},\{\{\mathtt{p}\}\})$ *does not satisfy* φ *either since, for any interpretation pair* (M, N)*, each* $I \in M$ *must satisfy* $(\mathtt{p} \vee \mathtt{q})$*. One such example is, .e.g., the interpretation pair* $M_3 = (\{\{\mathtt{p}\},\{\mathtt{q}\}\},\{\{\mathtt{p}\}\})$*.*

We now define the preference relation on MKNF interpretation pairs that is required for the notion of (non-monotonic) three-valued MKNF models, following an approach similar to the one in the two-valued case, i.e., by minimizing non-falsity (truth or undefinedness, in this case, of formulas w.r.t. **K**.

Definition 5.6. *Any MKNF interpretation pair* (M, N) *is a* three-valued MKNF model *for a given closed MKNF formula* φ *if*

(1) (M, N) *satisfies* φ *and*

(2) for each MKNF interpretation pair (M', N') *with* $M \subseteq M'$ *and* $N \subseteq N'$*, where at least one of the inclusions is proper and* $M' = N'$ *if* $M = N$*, there is* $I' \in M'$ *such that* $(I', \langle M', N' \rangle, \langle M, N \rangle)(\varphi) \neq \mathbf{t}$.

Condition (1) checks whether (M, N) evaluates φ to $\mathbf{t}$ while the second condition verifies that (M, N) contains only knowledge necessary to obtain this evaluation to $\mathbf{t}$. This is achieved by generalizing the corresponding notion in the two-valued MKNF semantics to the three-valued case: for each MKNF interpretation pair (M', N') that properly subsumes (M, N), it is checked that φ does not evaluate to $\mathbf{t}$ for all $I' \in M'$, where (M', N') is used to evaluate $\mathbf{K}$ while (M, N) evaluates **not**. Intuitively, one may consider an MKNF interpretation pair as a guess for the true evaluation of the considered formula, and condition (2) checks, having fixed the evaluation of modal **not**-atoms, whether the evaluation of modal $\mathbf{K}$-atoms is actually minimal w.r.t. to the order $\mathbf{f} < \mathbf{u} < \mathbf{t}$ of truth values. We illustrate in the example below how this minimization is achieved.

Example 5.7. *Consider the MKNF formula* φ *(corresponding to two MKNF rules):*

$$(\mathbf{not}\ \mathtt{p} \supset \mathbf{K}\ \mathtt{q}) \wedge (\mathbf{not}\ \mathtt{q} \supset \mathbf{K}\ \mathtt{p}) \tag{5.1}$$

Any MKNF interpretation pair (M, N) *that satisfies condition (1) of Definition 5.6 has to evaluate both conjuncts to true. For example, the MKNF interpretation pair* $(M, N) = (\{\{p\}, \{\mathtt{p}, \mathtt{q}\}\}, \{\{\mathtt{p}, \mathtt{q}\}\})$ *that evaluates* $\mathbf{K}\ \mathtt{p}$ *to* $\mathbf{t}$ *and* $\mathbf{K}\ \mathtt{q}$ *to* $\mathbf{u}$ *satisfies the first condition but is not a three-valued MKNF model of* φ *since, e.g.,* $(M', N') = (\{\emptyset, \{\mathtt{p}\}, \{\mathtt{q}\}, \{\mathtt{p}, \mathtt{q}\}\}, \{\{\mathtt{p}, \mathtt{q}\}\})$ *violates condition (2). In fact, this MKNF interpretation pair* (M', N') *is a three-valued MKNF model. The operator* **not** *is always evaluated w.r.t. the MKNF interpretation pair* (M, N)*, even when considering condition (2) of Definition 5.6, so, for* $N = \{\{\mathtt{p}, \mathtt{q}\}\}$*, the two implications are true anyway, and* M *has*

to be the set of all possible interpretations $\{\emptyset, \{\mathtt{p}\}, \{\mathtt{q}\}, \{\mathtt{p}, \mathtt{q}\}\}$ *to satisfy condition (2). Thus, we obtain the MKNF interpretation pair that evaluates* $\mathbf{K}\,\mathtt{p}$ *and* $\mathbf{K}\,\mathtt{q}$ *to* $\mathbf{u}$*. In other words, the initial MKNF interpretation pair was not minimal w.r.t. the evaluation of modal* $\mathbf{K}$*-atoms. Similar to the minimization of the evaluation of* $\mathbf{K}\,\mathtt{p}$ *from* $\mathbf{t}$ *to* $\mathbf{u}$*, changes from* $\mathbf{u}$ *to* $\mathbf{f}$ *are possible: maintain the original* $M = \{\{\mathtt{p}\}, \{\mathtt{p}, \mathtt{q}\}\}$ *and set* $N = M$*. Now the evaluation of* $\mathbf{K}\,\mathtt{q}$ *is minimized from* $\mathbf{u}$ *to* $\mathbf{f}$*, and it is easy to verify that the resulting MKNF interpretation pair is in fact a three-valued MKNF model of* φ*.*

It should be pointed out that the larger the set M or N is, the less true or undefined knowledge is inferred. So, minimization is achieved by increasing the sets in consideration. Note that $N' \subseteq M'$ for MKNF interpretation pairs (M', N') ensures that we only check reasonable candidates for augmenting (M, N).[1]

We continue Example 5.5 to exemplify these notions.

Example 5.8. *Consider again the formula* $\varphi = (\mathtt{p} \vee \mathtt{q}) \wedge \mathbf{not}\,\mathtt{q}$*, for which we know that* $M_3 = (\{\{\mathtt{p}\}, \{\mathtt{q}\}\}, \{\{\mathtt{p}\}\})$ *satisfies it. Clearly, no element of* φ *should be undefined, so any three-valued MKNF model should be total. Consequently,* M_3 *is no three-valued MKNF model. Instead, we have to find a maximal set* M *whose elements all model* $\mathtt{p} \vee \mathtt{q}$ *such that at least one does not model* $\mathtt{q}$*. Thus, the only three-valued MKNF model is* $M_4 = (\{\{\mathtt{p}\}, \{\mathtt{q}\}, \{\mathtt{p}, \mathtt{q}\}\}, \{\{\mathtt{p}\}, \{\mathtt{q}\}, \{\mathtt{p}, \mathtt{q}\}\})$*.*

The notions of consistency and entailment can straightforwardly be adapted.

Definition 5.9. *If there is a three-valued MKNF model for a given closed MKNF formula* φ*, then* φ *is called* MKNF-consistent*, otherwise* φ *is called* MKNF-inconsistent*. If* $(I, \langle M, N\rangle, \langle M, N\rangle)(\psi) = \mathbf{t}$ *for all three-valued MKNF models* (M, N) *of* φ*, then* φ entails ψ*, written* $\varphi \models^3_{MKNF} \psi$*.*

Note that MKNF-inconsistent MKNF formulas do not necessarily evaluate to $\mathbf{f}$. For example, $\varphi = \mathbf{K}\,\mathtt{u} \wedge \mathbf{not}\,\mathtt{u}$ evaluates to $\mathbf{u}$ for the MKNF interpretation pair $(\{\{\mathtt{u}\}, \emptyset\}, \{\{\mathtt{u}\}\})$. In fact, an MKNF-inconsistent formula can even evaluate to $\mathbf{t}$ and that was already the case for the two-valued MKNF semantics of [Motik and Rosati,

[1] In comparison to [Knorr et al., 2008] the definition has been slightly altered to simplify proofs and computation: in case of a total MKNF interpretation pair (M, M), it is sufficient to check that no other total MKNF interpretation pair (M', M') actually yields a true evaluation for all $I' \in M'$. This simplification is also justified by the intuition of enlarging N' separately: there is no undefinedness in a total MKNF interpretation pair, and minimization of undefinedness is thus not necessary.

2010]. E.g., $\varphi = \neg\mathbf{not}\,\mathtt{p}$ is MKNF-inconsistent, and it evaluates to $\mathbf{t}$ in some MKNF interpretation pairs (and also in some MKNF interpretations of [Motik and Rosati, 2010]). This does not constitute a problem since it does not affect the definitions of MKNF models or MKNF-consistency.

Though the notions of inconsistency and unsatisfiability are usually applied in the same technical sense, we want to distinguish between MKNF-satisfiability in the two-valued case and MKNF-consistency for three-valued MKNF models. Likewise, we distinguish between the two-valued notion 'MKNF entails' and the three-valued 'entails'. This avoids cumbersome notions such 'three-valued MKNF entails' but overloads the first-order notion 'entails'. It shall be obvious from the context, which of the two is meant.

5.3 General Properties of three-valued MKNF

Even though several notions of (two-valued) MKNF logics were adapted to the three-valued case, quite a few similarities exist. In particular, the next two properties of MKNF carry over to the three-valued case, whose original proofs can be found in [Motik and Rosati, 2006] and are adapted in the following.

The first property states that $\mathbf{K}$ can be introduced in front of an arbitrary closed MKNF formula φ without changing the three-valued MKNF models of φ.

Proposition 5.10. *Let σ be a closed MKNF formula and (M, N) an MKNF interpretation pair. Then, (M, N) is a three-valued MKNF model of σ if and only if (M, N) is a three-valued MKNF model of $\mathbf{K}\,\sigma$.*

Proof. Suppose that (M, N) is a three-valued MKNF model of the MKNF formula σ. We know for all $I \in M$ that $(I, \langle M, N\rangle, \langle M, N\rangle)(\sigma) = \mathbf{t}$. So $(I, \langle M, N\rangle, \langle M, N\rangle)(\mathbf{K}\,\sigma) = \mathbf{t}$ holds for all $I \in M$ as well. Since for each (M', N') there is an $I' \in M'$ such that $(I', \langle M', N'\rangle, \langle M, N\rangle)(\sigma) \neq \mathbf{t}$, we also obtain the same for $\mathbf{K}\,\sigma$, and (M, N) is a three-valued MKNF model of $\mathbf{K}\,\sigma$.

The converse direction follows in an analogous fashion. □

This allows us to conclude that, e.g., some formula σ can be entailed from some MKNF formula φ if and only if $\mathbf{K}\,\sigma$ can be entailed from that MKNF formula φ.

The second property we adapt from the two-valued to the three-valued MKNF semantics is also related to entailment. We show that grounding a hybrid MKNF

knowledge base $\mathcal{K}$ does not affect the three-valued MKNF models of $\mathcal{K}$. This shows that $\mathcal{K}$ and $\mathcal{K}_G$ derive exactly the same consequences.

Lemma 5.11. *Let $\mathcal{K}$ be a hybrid MKNF knowledge base and ψ a ground MKNF formula. Then $\mathcal{K} \models^3_{MKNF} \psi$ if and only if $\mathcal{K}_G \models^3_{MKNF} \psi$.*

Proof. The argument showing the contrapositive statement $\mathcal{K} \not\models^3_{MKNF} \psi$ if and only if $\mathcal{K}_G \not\models^3_{MKNF} \psi$ is absolutely identical to the one in [Motik and Rosati, 2006]. So we simply refer to the proof given there. □

This result is of course not restricted to hybrid MKNF knowledge bases, but this is the only case for which we specifically defined a ground form, and the only kind of MKNF formula we are interested in Chapter 6. More importantly, hybrid MKNF knowledge bases are implicitly admissible and therefore also DL-safe. Thus, grounding and reasoning in rules is restricted to the constants appearing in the knowledge base. A more general statement of that lemma for arbitrary formulas would not hold since reasoning would no longer be decidable.

5.4 A Well-Founded MKNF Model

MKNF interpretation pairs, and therefore also three-valued MKNF models, can be compared by an order that resembles the knowledge order from Logic Programming (see Definition 2.23). Intuitively, given such an order and two MKNF interpretation pairs (M_1, N_1) and (M_2, N_2), we have that (M_1, N_1) is greater than (M_2, N_2) w.r.t. such an order if (M_1, N_1) allows us to derive more true and false knowledge than (M_2, N_2). Taking into account that a larger set of interpretations derives less true and more false knowledge, we can define an order on MKNF interpretation pairs.

Definition 5.12. *Let (M_1, N_1) and (M_2, N_2) be MKNF interpretation pairs. We have that $(M_1, N_1) \succeq_k (M_2, N_2)$ iff $M_1 \subseteq M_2$ and $N_1 \supseteq N_2$.*

Example 5.13. *Consider $(M_1, N_1) = (\{\{\mathtt{p}\}, \{\mathtt{q}\}\}, \{\{\mathtt{p}\}, \{\mathtt{q}\}, \{\mathtt{p}, \mathtt{q}\}, \emptyset\})$ and $(M_2, N_2) = (\{\{\mathtt{p}\}, \{\mathtt{q}\}\}, \{\{\mathtt{p}\}, \{\mathtt{q}\}\})$. Then, we have that $(M_1, N_1) \succeq_k (M_2, N_2)$.*

Such an order is of particular interest for comparing models. In Logic Programming the least model w.r.t. derivable knowledge among all three-valued models for a given

program is the well-founded model. Here, we want to introduce a similar notion referring to the minimal three-valued MKNF models, i.e., the ones among all three-valued MKNF models that leave as much as possible undefined.

Definition 5.14. *Let φ be a closed MKNF formula and (M, N) a three-valued MKNF model of φ such that $(M_1, N_1) \succeq_k (M, N)$ for all three-valued MKNF models (M_1, N_1) of φ. Then (M, N) is a* well-founded MKNF model *of φ.*

Of course, if φ is inconsistent, then there are no three-valued MKNF models and thus no well-founded MKNF models of φ. However, if φ is a consistent hybrid MKNF knowledge base, it is guaranteed that a well-founded MKNF model of φ exists. Moreover, this well-founded model is unique. As we shall see, this model is especially important in that a modal atom $\mathbf{K}\,H$ is true in the well-founded MKNF model iff $\mathbf{K}\,H$ is true in all three-valued MKNF models. This way, performing skeptical reasoning in three-valued MKNF models amounts to determining the well-founded MKNF model.

Theorem 5.15. *If $\mathcal{K}$ is an MKNF-consistent hybrid MKNF KB, then a well-founded MKNF model exists, and it is unique.*

The respective proofs for the uniqueness/existence of the well-founded MKNF model, and how to calculate this unique model, are presented in Chapter 6 (as a direct consequence of Theorem 6.45).

The following example gives at least an intuitive insight into the correspondence between two-valued and three-valued MKNF models, and the well-founded MKNF model.

Example 5.16. *Consider the knowledge base $\mathcal{K}$ corresponding to the MKNF formula φ from Example 5.7.*

$$\mathbf{K}\,\mathtt{q} \leftarrow \mathbf{not}\,\mathtt{p} \tag{5.2}$$

$$\mathbf{K}\,\mathtt{p} \leftarrow \mathbf{not}\,\mathtt{q} \tag{5.3}$$

The two-valued MKNF models of $\mathcal{K}$ are $\{\{\mathtt{p}\}, \{\mathtt{p}, \mathtt{q}\}\}$ and $\{\{\mathtt{q}\}, \{\mathtt{p}, \mathtt{q}\}\}$, i.e., $\mathbf{K}\,\mathtt{p}$ and **not** $\mathtt{q}$ *are true in the first model, and $\mathbf{K}\,\mathtt{q}$ and* **not** $\mathtt{p}$ *are true in the second one. Given these two two-valued MKNF models, we can obtain two total three-valued MKNF models: $(\{\{\mathtt{p}\}, \{\mathtt{p}, \mathtt{q}\}\}, \{\{\mathtt{p}\}, \{\mathtt{p}, \mathtt{q}\}\})$ and $(\{\{\mathtt{q}\}, \{\mathtt{p}, \mathtt{q}\}\}, \{\{\mathtt{q}\}, \{\mathtt{p}, \mathtt{q}\}\})$. As we have already seen in Example 5.7, the only other three-valued MKNF model of $\mathcal{K}$ is $M =$*

$(\{\emptyset, \{\mathtt{p}\}, \{\mathtt{q}\}, \{\mathtt{p}, \mathtt{q}\}\}, \{\{\mathtt{p}, \mathtt{q}\}\})$. *This MKNF model satisfies the condition given in Definition 5.14, and M is thus a well-founded MKNF model of $\mathcal{K}$. In fact, M is the only well-founded MKNF model.*

5.5 Relation with the Two-valued MKNF Semantics

As already pointed out, two- and three-valued MKNF models are closely related and the correspondence in the previous example is no coincidence. We therefore now show that any two-valued MKNF model M corresponds exactly to a (total) three-valued MKNF model (M, M) and vice versa.

For that purpose, we first prove that the evaluation in an MKNF structure (I, M, N) and the evaluation in a total three-valued structure $(I, \langle M, M\rangle, \langle N, N\rangle)$ are identical. Intuitively, this holds because nothing can be undefined in a total three-valued structure.

Lemma 5.17. *Let φ be a closed MKNF formula. Then we have $(I, M, N) \models \varphi$ if and only if $(I, \langle M, M\rangle, \langle N, N\rangle)(\varphi) = \mathbf{t}$.*

Proof. The proof is done by induction on the formula φ.

Let φ be $P(t_1, \ldots, t_n)$. We have $(I, M, N) \models P(t_1, \ldots, t_n)$ iff $(t_1^I, \ldots, t_n^I) \in P^I$ iff $(I, \langle M, M\rangle, \langle N, N\rangle)(P(t_1, \ldots, t_n)) = \mathbf{t}$.

Assume that the lemma holds for φ_1. We show the induction steps for $\neg$ and $\mathbf{K}$, all the other cases follow analogously.

Let φ be $\neg\varphi_1$. We have that $(I, M, N) \models \neg\varphi_1$ iff $(I, M, N) \not\models \varphi_1$ iff, by the induction hypothesis, $(I, \langle M, M\rangle, \langle N, N\rangle)(\varphi_1) = \mathbf{f}$ iff by definition of evaluation in three-valued MKNF structures $(I, \langle M, M\rangle, \langle N, N\rangle)(\neg\varphi_1) = \mathbf{t}$.

Let φ be $\mathbf{K}\,\varphi_1$. We have $(I, M, N) \models \mathbf{K}\,\varphi_1$ iff $(I, M, N) \models \varphi_1$ holds for each $I \in M$ iff $(I, \langle M, M\rangle, \langle N, N\rangle)(\varphi_1) = \mathbf{t}$ for all $I \in M$ by the induction hypothesis iff $(I, \langle M, M\rangle, \langle N, N\rangle)(\mathbf{K}\,\varphi_1) = \mathbf{t}$. □

This lemma can be used to show that every two-valued MKNF model M corresponds to a three-valued MKNF model (M, M), and also the converse, i.e., that every three-valued MKNF model (M, M) corresponds to a two-valued MKNF model in the sense of [Motik and Rosati, 2010].

Proposition 5.18. *Given a closed MKNF formula φ, M is a two-valued MKNF model of φ if and only if (M, M) is a three-valued MKNF model of φ.*

Proof. Let (M, M) be a three-valued MKNF model of φ, i.e., (M, M) satisfies the two conditions of Definition 5.6. We show that M is a two-valued MKNF model of φ. It follows from the first of the two conditions of Definition 5.6 that $(I, \langle M, M\rangle, \langle M, M\rangle)(\varphi) = \mathbf{t}$ for all $I \in M$ and therefore, by Lemma 5.17, that $(I, M, M) \models \varphi$ for each $I \in M$. The second condition states, for each MKNF interpretation pair (M', M') with $M \subset M'$, that we have $(I', \langle M', M'\rangle, \langle M, M\rangle)(\varphi) \neq \mathbf{t}$ for some $I' \in M'$. We conclude from Lemma 5.17 that for any M' with $M' \supset M$ there is an $I' \in M'$ such that $(I', M', M) \not\models \varphi$.

Now, let M be a two-valued MKNF model of φ. We show that (M, M) is a three-valued MKNF model of φ. We know that $(I, M, M) \models \varphi$ for each $I \in M$ since M is a two-valued MKNF model of φ. As such, $(I, \langle M, M\rangle, \langle M, M\rangle)(\varphi) = \mathbf{t}$ holds for all $I \in M$ by Lemma 5.17, and so the first of the two conditions of Definition 5.6 is satisfied. Furthermore, since M is a two-valued MKNF model of φ, we know that for all M' with $M' \supset M$ we have $(I', M', M) \not\models \varphi$ for some $I' \in M'$. Again, from Lemma 5.17, we know that for any MKNF interpretation pair (M', M') with $M' \supset M$ we have $(I', \langle M', M'\rangle, \langle M, M\rangle)(\varphi) \neq \mathbf{t}$ for some $I' \in M'$. This is sufficient since, according to Definition 5.6, for (M, M) we only need to consider total MKNF interpretation pairs (M', M'). □

We can conclude that our semantics is an extension of the two-valued MKNF semantics, in the sense that each two-valued model is also a (corresponding) model in our semantics. Several properties of the original semantics, such as non-monotonicity, or the properties in Section 5.3, hold as well, and the intuitive ideas based on which the new semantics is defined follow the intuition of the two-valued case. In particular, in both cases the semantics is based on a monotonic evaluation and a non-monotonic semantics is obtained by minimizing true (and undefined) knowledge.

We have thus defined the first three-valued MKNF semantics, which also allows us to extend the capability of MKNF to be used as a unifying framework for non-monotonic reasoning formalisms. We discuss this aspect in more detail in Chapter 7.

Finally, we want to point out that our proposal is by no means the only possible one for a three-valued MKNF semantics. We may alter, e.g., the evaluation of undefinedness w.r.t. the operator $\supset$ and obtain a different semantics. Here, however, since we aim at a semantics that is faithful w.r.t. the Well-Founded Semantics of LP, we claim that our choice is the appropriate one for our intentions. This is shown in Chapter 6, where we define how to compute a well-founded model for hybrid MKNF knowledge bases.

6

Alternating Fixpoint for the Well-founded MKNF Model

In this chapter we prove that the well-founded MKNF model is unique, and we define a procedure for computing this unique model. For that purpose, the alternating fixpoint construction of [Hitzler and Wendt, 2005; van Gelder, 1989] for the well-founded semantics of logic programs is adapted to hybrid MKNF knowledge bases, taking into account possible conflicts resulting from the combination of classical negation in ontologies and non-monotonic negation in rules.

In more detail, we are going to adapt a finite representation of MKNF interpretations to MKNF interpretation pairs that resembles working with Herbrand bases in Logic Programming. Then, we take the operator defined for hybrid MKNF knowledge bases with definite MKNF rules from [Motik and Rosati, 2010] and use it to define a stable condition for hybrid MKNF knowledge bases with normal rules. Based on this, we define two interacting operators that provide two fixed points, from which we are able to derive a representation that corresponds to an MKNF interpretation pair, and that is precisely the unique well-founded MKNF model of the KB in consideration if it is MKNF-consistent. MKNF-consistency can be verified in a straightforward way and several desirable key properties are proven in this chapter.

We start by adapting partitions from [Motik and Rosati, 2010] as the means of representing MKNF interpretation pairs and show that such a representation, indeed, does correspond to the respective MKNF interpretation pair or three-valued MKNF model (Section 6.1). Then, based on that representation, we define operators that allow

us to compute the representation of a unique model for hybrid MKNF knowledge bases (Section 6.2). We devise an alternative characterization based on unfounded sets (as known from logic programs) and show that the computed result of that characterization coincides with that representation of the unique model (Section 6.3). We show that this model is indeed the (unique) well-founded MKNF model, and we present several important properties including the computational complexity, faithfulness w.r.t. the well-founded semantics of logic programs, and discovery of inconsistencies (Section 6.4).

6.1 Partitions of Modal Atoms

As argued in [Motik and Rosati, 2010], since there are infinitely many two-valued MKNF models of an arbitrary hybrid MKNF knowledge base with a countably infinite domain, working directly with two-valued MKNF models is cumbersome. In fact, even in case of simple finite two-valued MKNF models, the representation is rather space-consuming already and this holds all the more for MKNF interpretation pairs in the three-valued MKNF semantics presented in Chapter 5 (see, e.g. Example 5.8). So, some finite, compact representation is required. The solution, applied in [Motik and Rosati, 2010] and originally from [Donini et al., 2002], is to represent a two-valued MKNF model by a finite first-order formula whose set of (first-order) models corresponds to the two-valued MKNF model itself. Intuitively, such a first-order formula is obtained in [Motik and Rosati, 2010] by first dividing the modal atoms occurring in the ground hybrid MKNF knowledge base into true and false (modal) **K**-atoms, and then constructing the first-order formula from the true **K**-atoms and the ontology. We extend this construction, and the related notions from [Motik and Rosati, 2010], to three truth values by partitioning atoms into three sets.

Definition 6.1. *Let $\mathcal{K}_G = (\mathcal{O}, \mathcal{P}_G)$ be a ground hybrid MKNF knowledge base. The* set of **K**-atoms of $\mathcal{K}_G$*, written* $\mathsf{KA}(\mathcal{K}_G)$*, is the smallest set that contains (i) all ground* **K***-atoms occurring in $\mathcal{P}_G$, and (ii) a* **K***-atom* $\mathbf{K}\,\xi$ *for each ground* **not***-atom* $\mathbf{not}\,\xi$ *occurring in $\mathcal{P}_G$. A* partial partition (T, F) *of* $\mathsf{KA}(\mathcal{K}_G)$ *consists of two sets, where $T, F \subseteq \mathsf{KA}(\mathcal{K}_G)$ and $T \cap F = \emptyset$. A third set U is implicitly defined as $\mathsf{KA}(\mathcal{K}_G) \setminus (T \cup F)$.*

The set $\mathsf{KA}(\mathcal{K}_G)$ contains all modal atoms occurring in $\mathcal{K}_G$, only with **not**-atoms substituted by corresponding modal **K**-atoms. This set is partitioned into three sets T, F, and U where, intuitively, T contains true **K**-atoms, F contains false **K**-atoms, and

U contains all the remaining that are considered to be undefined. Clearly, a **not**-atom **not** A is intended to be true if $\mathbf{K}\,A$ is false, it is intended to be false if $\mathbf{K}\,A$ is true, and undefined otherwise.

Example 6.2. *Consider the simple, already ground hybrid MKNF knowledge base $\mathcal{K}_G$.*

$$\mathtt{Q} \sqsubseteq \mathtt{R} \tag{6.1}$$

$$\mathbf{K}\,\mathtt{p} \leftarrow \mathbf{not}\,\mathtt{p} \tag{6.2}$$

$$\mathbf{K}\,\mathtt{Q}(\mathtt{a}) \leftarrow \mathbf{K}\,\mathtt{R}(\mathtt{a}) \tag{6.3}$$

$$\mathbf{K}\,\mathtt{s} \leftarrow \mathbf{not}\,\mathtt{Q}(\mathtt{a}) \tag{6.4}$$

We have that $\mathsf{KA}(\mathcal{K}_G) = \{\mathbf{K}\,\mathtt{p}, \mathbf{K}\,\mathtt{Q}(\mathtt{a}), \mathbf{K}\,\mathtt{R}(\mathtt{a}), \mathbf{K}\,\mathtt{s}\}$. *One possible partial partition for that set of* $\mathbf{K}$*-atoms is* $(T, F) = (\{\mathbf{K}\,\mathtt{s}\}, \{\mathbf{K}\,\mathtt{Q}(\mathtt{a}), \mathbf{K}\,\mathtt{R}(\mathtt{a})\})$ *with the intention that* $\mathbf{K}\,\mathtt{s}$ *is true, that* $\mathbf{K}\,\mathtt{p}$ *is undefined, and that the other two* $\mathbf{K}$*-atoms are false.*

In [Motik and Rosati, 2010], given a knowledge base $\mathcal{K}_G$, a set of first-order formulas is defined with the aim of using the models of this set of formulas to represent the models of both the ontology and a set of true modal atoms. If this set of true modal atoms is properly chosen, then the set of first-order interpretations satisfying that set of formulas corresponds to one two-valued MKNF model of $\mathcal{K}_G$.

Here, this construction does not suffice, and we show how to adapt the idea to a three-valued setting. For that, the definition of the set of first-order formulas that is intended to represent the KB can be recalled from [Motik and Rosati, 2010].

Definition 6.3. *Let $\mathcal{K}_G = (\mathcal{O}, \mathcal{P}_G)$ be a ground hybrid MKNF knowledge base. For a subset S of $\mathsf{KA}(\mathcal{K}_G)$, the* objective knowledge *of S w.r.t. $\mathcal{K}_G$ is the set of first-order formulas $\mathsf{OB}_{\mathcal{O},S} = \{\pi(\mathcal{O})\} \cup \{\xi \mid \mathbf{K}\,\xi \in S\}$.*

This notion is used below to establish a link between three-valued MKNF models and partial partitions. In fact, in the two-valued case, S is simply the set of true $\mathbf{K}$-atoms.

Example 6.4. *Recall Example 6.2. The objective knowledge for the partial partition (T, F) presented could be defined using the true $\mathbf{K}$-atoms just as in the two-valued case. Then, $\mathsf{OB}_{\mathcal{O},T} = \{\pi(\{\mathtt{Q} \sqsubseteq \mathtt{R}\})\} \cup \{\mathtt{s}\}$. Clearly, in this representation, the fact that $\mathbf{K}\,\mathtt{p}$ is supposed to be undefined is lost, which is no surprise given that undefined modal atoms do not occur in the two-valued case.*

We obtain from that example that we need to define the correspondence in a different manner. But for this purpose, we need to adapt one more notion from [Motik and Rosati, 2010].

Definition 6.5. *Let S be a set of ground* $\mathbf{K}$*-atoms. The partial partition (T, F) of S is* induced by *an MKNF interpretation pair (M, N) as follows:*

(1) $\mathbf{K}\,\xi \in T$ *implies* $\forall I \in M : (I, \langle M, N\rangle, \langle M, N\rangle)(\mathbf{K}\,\xi) = \mathbf{t}$,

(2) $\mathbf{K}\,\xi \in F$ *implies* $\forall I \in M : (I, \langle M, N\rangle, \langle M, N\rangle)(\mathbf{K}\,\xi) = \mathbf{f}$, *and*

(3) $\mathbf{K}\,\xi \notin T$ *and* $\mathbf{K}\,\xi \notin F$ *implies* $\forall I \in M : (I, \langle M, N\rangle, \langle M, N\rangle)(\mathbf{K}\,\xi) = \mathbf{u}$.

This relation formally establishes the 'intended semantics' of partial partitions. Any given MKNF interpretation pair (M, N), and in particular also a three-valued MKNF model, induces a partition (T, F) such that all elements of T are true in (M, N), all elements of F are false in (M, N), and the remaining are undefined.

Example 6.6. *Recall Example 6.2 and consider the three-valued MKNF model* $(M, N) = (\{\{\mathtt{s}\}, \{\mathtt{s}, \mathtt{p}\}, \{\mathtt{s}, \mathtt{p}, \mathtt{Q}(\mathtt{a}), \mathtt{R}(\mathtt{a})\}\}, \{\{\mathtt{s}, \mathtt{p}\}, \{\mathtt{s}, \mathtt{p}, \mathtt{Q}(\mathtt{a}), \mathtt{R}(\mathtt{a})\}\})$*. Then (M, N) does precisely induce the partition (T, F) given in Example 6.2. Of course, (M, N) is not the only MKNF interpretation pair that induces that partition, but it is in fact a three-valued MKNF model of $\mathcal{K}_G$. Intuitively, that model can also be obtained by setting M to model $\pi(\mathcal{O})$ and T and N to model $\pi(\mathcal{O})$ and everything that is not contained in F. Formally, we would obtain* $M = \{I \mid I \models ((\forall \mathtt{x} : \mathtt{Q}(\mathtt{x}) \supset \mathtt{R}(\mathtt{x})) \wedge \mathtt{s})\}$ *and* $N = \{I \mid I \models ((\forall \mathtt{x} : \mathtt{Q}(\mathtt{x}) \supset \mathtt{R}(\mathtt{x})) \wedge (\mathtt{s} \wedge \mathtt{p}))\}$.

Based on this relation between partial partitions and MKNF interpretation pairs, we can show that the objective knowledge derived from the partial partition that is induced by a three-valued MKNF model is identical to that model, following the construction sketched in the example above. This result is used in Section 6.4 to show that the specific partition we compute produces a three-valued MKNF model (Theorem 6.44).

Proposition 6.7. *Let (M, N) be a three-valued MKNF model of a ground hybrid MKNF knowledge base $\mathcal{K}_G = (\mathcal{O}, \mathcal{P}_G)$, and (T, F) the partition of $\mathsf{KA}(\mathcal{K}_G)$ induced by (M, N). Then $(M, N) = (\{I \mid I \models \mathsf{OB}_{\mathcal{O},T}\}, \{I \mid I \models \mathsf{OB}_{\mathcal{O},\mathsf{KA}(\mathcal{K}_G)\setminus F}\})$.*

Proof. For $\mathcal{K}_G = (\mathcal{O}, \mathcal{P}_G)$ a ground hybrid MKNF knowledge base, let (M, N) be a three-valued MKNF model of $\mathcal{K}_G$, (T, F) the partition of $\mathsf{KA}(\mathcal{K}_G)$ induced by (M, N),

and $(M', N') = (\{I \mid I \models \mathsf{OB}_{\mathcal{O},T}\}, \{I \mid I \models \mathsf{OB}_{\mathcal{O},\mathsf{KA}(\mathcal{K}_G)\setminus F}\})$. We show that $(M, N) = (M', N')$.

First, we show that $M \subseteq M'$. Let I be an interpretation in M. We show that $I \in M' = \{I \mid I \models \mathsf{OB}_{\mathcal{O},T}\}$, i.e., that $I \models \{\pi(\mathcal{O})\} \cup \{\xi \mid \mathbf{K}\,\xi \in T\}$. Since (M, N) is a three-valued MKNF model of $\mathcal{K}_G$, we know that $(M, N) \models \mathbf{K}\,\pi(\mathcal{O})$. Thus, we have $I \models \pi(\mathcal{O})$. Consider each $\mathbf{K}\,\xi \in T$. Since (M, N) induces the partition (T, F) we have $(I, \langle M, N\rangle, \langle M, N\rangle)(\mathbf{K}\,\xi) = \mathbf{t}$ and thus $I \models \xi$. Hence, $I \models \mathsf{OB}_{\mathcal{O},T}$. Consequently, $I \in M'$ holds and therefore $M \subseteq M'$.

Next, we show that $N \subseteq N'$. Let I be an interpretation in N. We show that $I \in N' = \{I \mid I \models \mathsf{OB}_{\mathcal{O},\mathsf{KA}(\mathcal{K}_G)\setminus F}\}$, i.e., that $I \models \{\pi(\mathcal{O})\} \cup \{\xi \mid \mathbf{K}\,\xi \in \mathsf{KA}(\mathcal{K}_G) \setminus F\}$. We already know that, for each $I \in M$, $I \models \pi(\mathcal{O})$. Since $N \subseteq M$, we also have that $I \models \pi(\mathcal{O})$ for each $I \in N$. Consider each $\mathbf{K}\,\xi \notin F$. The premise of condition (2) in Definition 6.5 is false, but the premises of conditions (1) and (3) in that definition are true. We show for both cases that $I \models \xi$. This suffices to show that $I \in N'$, i.e., that $I \models \mathsf{OB}_{\mathcal{O},\mathsf{KA}(\mathcal{K}_G)\setminus F}$, which shows $N \subseteq N'$. In the case of (1), we already know that $I \models \xi$ for each $I \in M$, and, since $N \subseteq M$ holds, we also have $I \models \xi$ for each $I \in N$. In the case of (3), we know that $(I, \langle M, N\rangle, \langle M, N\rangle)(\mathbf{K}\,\xi) = \mathbf{u}$ for each $I \in M$. Thus, $I \models \xi$ holds for each $I \in N$.

We now show that each of the two sets are in fact identical, i.e., $M = M'$ and $N = N'$. Note first that $T \subseteq \mathsf{KA}(\mathcal{K}_G)\setminus F$. Thus, for any $I \in N'$, we have $I \in \{I \mid I \models \mathsf{OB}_{\mathcal{O},T}\}$ and therefore $N' \subseteq M'$, i.e., (M', N') is an MKNF interpretation pair. So assume that (M', N') is an MKNF interpretation pair with $M \subseteq M'$ and $N \subseteq N'$, where at least one of the inclusions is proper. We show that $(I', \langle M', N'\rangle, \langle M, N\rangle)(\mathcal{K}_G) = \mathbf{t}$ for all $I' \in M'$, and we thus derive a contradiction to (M, N) being a three-valued MKNF model of $\mathcal{K}_G$. For the former, it suffices to prove that $(I', \langle M', N'\rangle, \langle M, N\rangle)(\mathbf{K}\,\pi(\mathcal{O}) \wedge \pi(\mathcal{P}_G)) = \mathbf{t}$ for all $I' \in M'$. By definition of M' we know that $(I', \langle M', N'\rangle, \langle M, N\rangle)(\mathbf{K}\,\pi(\mathcal{O})) = \mathbf{t}$ for all $I' \in M'$. We only have to show the same for $\pi(\mathcal{P}_G)$. We achieve that by showing that, for each case of Definition 6.5, the modal atoms appearing in $\pi(\mathcal{P}_G)$ are evaluated to identical truth values in (M, N) and (M', N'). This suffices to show that $(I', \langle M', N'\rangle, \langle M, N\rangle)(\pi(\mathcal{P}_G)) = \mathbf{t}$ for all $I' \in M'$ since (M, N), as a three-valued MKNF model of $\mathcal{K}_G$, ensures that $(I', \langle M, N\rangle, \langle M, N\rangle)(\pi(\mathcal{P}_G)) = \mathbf{t}$. We thus obtain a contradiction to (M, N) being a three-valued MKNF model.

- Consider each $\mathbf{K}\,\xi \in T$. We obtain $(I', \langle M', N'\rangle, \langle M, N\rangle)(\mathbf{K}\,\xi) = \mathbf{t}$ for all $I' \in M'$ by definition of M' just as we have $(I, \langle M, N\rangle, \langle M, N\rangle)(\mathbf{K}\,\xi) = \mathbf{t}$ for all $I \in M$ by Definition 6.5.

- Consider each $\mathbf{K}\,\xi \in F$. We obtain $(I, \langle M, N\rangle, \langle M, N\rangle)(\mathbf{K}\,\xi) = \mathbf{f}$, by Definition 6.5. We derive that, by Definition 5.2, $(I, \langle M, N\rangle, \langle M, N\rangle)(\mathbf{K}\,\xi) = \mathbf{f}$ for some $I \in N$. Because of that, and since $N \subseteq N'$, we also have $(I', \langle M', N'\rangle, \langle M, N\rangle)(\mathbf{K}\,\xi) = \mathbf{f}$ for some $I' \in N'$. Thus, by Definition 5.2, $(I, \langle M', N'\rangle, \langle M, N\rangle)(\mathbf{K}\,\xi) = \mathbf{f}$.

- Consider each $\mathbf{K}\,\xi$ with $\mathbf{K}\,\xi \notin F$ and $\mathbf{K}\,\xi \notin T$. By Definition 6.5, we obtain $(I, \langle M, N\rangle, \langle M, N\rangle)(\mathbf{K}\,\xi) = \mathbf{u}$. By definition of N', $(I', \langle M', N'\rangle, \langle M, N\rangle)(\mathbf{K}\,\xi) \neq \mathbf{f}$ holds. From $(I, \langle M, N\rangle, \langle M, N\rangle)(\mathbf{K}\,\xi) = \mathbf{u}$ and $M \subseteq M'$ we conclude that only $(I', \langle M', N'\rangle, \langle M, N\rangle)(\mathbf{K}\,\xi) = \mathbf{u}$ is possible.

- Consider any modal **not**-atom appearing in $\pi(\mathcal{P}_G)$. Since the evaluation of these is done in both cases w.r.t. (M, N), we straightforwardly obtain the identical evaluation.

This finishes the proof. □

The next example illustrates the previously introduced notions using the knowledge base for recommending CDs.

Example 6.8. *Consider $\mathcal{K}$ consisting only of rule (4.8) from Example 4.17 and an ontology containing just one assertion:*

$$\mathbf{K}\,\mathtt{Recommend(x)} \leftarrow \mathbf{K}\,\mathtt{CD(x)}, \mathbf{not}\,\mathtt{owns(x)}, \mathbf{not}\,\mathtt{LowEval(x)}, \quad (6.5)$$
$$\mathbf{K}\,\mathtt{interesting(x)}.$$
$$\mathtt{CD(BNAW)} \quad (6.6)$$

The ground KB $\mathcal{K}_G$ contains one rule that results from (6.5) by substituting x with BNAW*. We thus obtain*

$$\mathsf{KA}(\mathcal{K}_G) = \{\mathbf{K}\,\mathtt{Recommend(BNAW)}, \mathbf{K}\,\mathtt{CD(BNAW)}, \mathbf{K}\,\mathtt{owns(BNAW)},$$
$$\mathbf{K}\,\mathtt{lowEval(BNAW)}, \mathbf{K}\,\mathtt{interesting(BNAW)}\}.$$

One can easily check that there is only one three-valued MKNF model (M, N) of $\mathcal{K}_G$, namely the one in which each $I \in M, N$ satisfies $I \models \mathtt{CD(BNAW)}$. This three-valued MKNF model induces the partition in which CD(BNAW) *appears in T and all other modal $\mathbf{K}$-atoms in F. The related set of first-order formulas just contains* CD(BNAW)*. This is reasonable since the ground version of (6.5) does not allow us to derive anything, and so we can ignore (6.5) when considering three-valued MKNF models of $\mathcal{K}_G$.*

Thus, we can guess the three-valued MKNF model and induce a partition or, alternatively, obtain the appropriate partial partition and the corresponding three-valued MKNF model from it. Now, we show how to compute a partial partition of a hybrid MKNF knowledge base.

6.2 Computation of the Alternating Fixpoint

As we have seen in Chapter 5, a knowledge base may in general have several three-valued MKNF models. But we have a special interest in the least one w.r.t. derivable knowledge – the well-founded MKNF model – and the computation of that model. In order to obtain the well-founded MKNF model and the corresponding partial partition, we resort to several existing relations and correspondences with semantics from Logic Programming.

The stable models of a normal logic program Π are the fixpoints of the Gelfond-Lifschitz operator Γ_Π [Gelfond and Lifschitz, 1988]. The same operator can be used to compute the (three-valued) well-founded model of Π by the so-called alternating fixpoint computation (cf. [van Gelder, 1989], see also Theorem 2.31). Intuitively, an operator, which results from applying Γ_Π twice, is used to compute a least and a greatest fixpoint, which correspond, respectively, to the true and non-false knowledge. The term "alternating" stems from the fact that Γ_Π is antitonic, and so successive applications, in turn, overestimate and underestimate derivable knowledge in the well-founded model, ultimately alternating between the two fixpoints. More precisely, by iteratively applying Γ_Π starting with an empty set of atoms, we first obtain a set of atoms that includes all the true atoms in the well-founded model of Π, i.e., an overestimate of the true atoms in the well-founded model (in other words, a set whose complement is an underestimate of the set of all false atoms). If we apply Γ_Π again to that result, then we obtain a set of atoms that are true for sure, i.e., an underestimate of the set of all true atoms. If we continue the iteration, we obtain alternating smaller overestimates and larger underestimates until eventually the iteration alternates between two fixpoints – one with all true atoms, and the other one with all atoms that are true or undefined in the well-founded model of Π.

Since stable models of logic programs and two-valued MKNF models are closely related, we adapt this scheme to hybrid MKNF knowledge bases. We define operators

that provide a stable condition for hybrid MKNF knowledge bases, and we use these operators to obtain an alternating fixpoint that corresponds to the well-founded MKNF model.

We start by defining an operator $T_{\mathcal{K}_G}$ that, given a set of **K**-atoms, draws conclusions from a positive ground hybrid MKNF knowledge base $\mathcal{K}_G$, i.e., a ground hybrid MKNF knowledge base where rules are of the form:

$$\mathbf{K}\, H \leftarrow \mathbf{K}\, A_1, \ldots, \mathbf{K}\, A_n \tag{6.7}$$

Definition 6.9. *Let $\mathcal{K}_G = (\mathcal{O}, \mathcal{P}_G)$ be a positive, ground hybrid MKNF knowledge base. The operators $R_{\mathcal{K}_G}$, $D_{\mathcal{K}_G}$, and $T_{\mathcal{K}_G}$ are defined on subsets of $\mathsf{KA}(\mathcal{K}_G)$ as follows:*

$$\begin{aligned} R_{\mathcal{K}_G}(S) = \{&\mathbf{K}\, H \mid \mathcal{P}_G \text{ contains a rule of the form (6.7)} \\ &\text{such that, for all } i,\ 1 \leq i \leq n, \mathbf{K}\, A_i \in S\} \end{aligned}$$

$$D_{\mathcal{K}_G}(S) = \{\mathbf{K}\, \xi \mid \mathbf{K}\, \xi \in \mathsf{KA}(\mathcal{K}_G) \text{ and } \mathsf{OB}_{\mathcal{O},S} \models \xi\}$$

$$T_{\mathcal{K}_G}(S) = R_{\mathcal{K}_G}(S) \cup D_{\mathcal{K}_G}(S)$$

Note that this operator is a slight variation of the one from [Motik and Rosati, 2010]. There it is verified in case of $R_{\mathcal{K}_G}$ whether all A_i follow from $\mathsf{OB}_{\mathcal{O},S}$, while we rely on the set S. Thus intermediate results slightly differ, but we resort to the definition presented here, because it allows us to define reasoning in rules in terms of rules alone (admitting derivations from the $\mathcal{O}$ only indirectly via S). This is also of advantage in Chapter 8 when defining a procedure that reverts the computation, i.e., one that allows queries instead of computing the whole model.

Example 6.10. *The knowledge base below exemplifies the difference of the two definitions.*

$$\mathtt{P} \sqsubseteq \mathtt{Q} \tag{6.8}$$

$$\mathbf{K}\, \mathtt{r} \leftarrow \mathbf{K}\, \mathtt{Q}(\mathtt{x}) \tag{6.9}$$

$$\mathbf{K}\, \mathtt{P}(\mathtt{a}) \leftarrow \tag{6.10}$$

Given $S = \{\mathbf{K}\, \mathtt{P}(\mathtt{a})\}$, we derive that $T_{\mathcal{K}_G}(S) = \{\mathbf{K}\, \mathtt{P}(\mathtt{a}), \mathbf{K}\, \mathtt{Q}(\mathtt{a})\}$ while the respective operator $T'_{\mathcal{K}_G}$ in [Motik and Rosati, 2010] derives that $T'_{\mathcal{K}_G}(S) = \{\mathbf{K}\, \mathtt{P}(\mathtt{a}), \mathbf{K}\, \mathtt{Q}(\mathtt{a}), \mathbf{K}\, \mathtt{r}\}$. This does not constitute a problem since taking the result $T_{\mathcal{K}_G}(S)$ and applying it again to $T_{\mathcal{K}_G}$ allows us to derive $\mathbf{K}\, \mathtt{r}$ as well.

The operator $R_{\mathcal{K}_G}$ derives immediate consequences from the rules in $\mathcal{K}_G$ while $D_{\mathcal{K}_G}$ yields consequences from the ontology combined with the already known information in S. The operator $T_{\mathcal{K}_G}$, which combines the other two, is monotonic:

Proposition 6.11. *Let $\mathcal{K}_G = (\mathcal{O}, \mathcal{P}_G)$ be a positive ground hybrid MKNF knowledge base, and $S \subseteq S' \subseteq \mathsf{KA}(\mathcal{K}_G)$. Then $T_{\mathcal{K}_G}(S) \subseteq T_{\mathcal{K}_G}(S')$.*

Proof. Suppose that $\mathbf{K}\,H \in T_{\mathcal{K}_G}(S)$. By Definition 6.9, $\mathbf{K}\,H \in R_{\mathcal{K}_G}(S) \cup D_{\mathcal{K}_G}(S)$ holds. If $\mathbf{K}\,H \in R_{\mathcal{K}_G}(S)$, then $\mathcal{P}_G$ contains a rule of the form (6.7) such that $\mathbf{K}\,A_i \in S$ for each $1 \leq i \leq n$. Since $S \subseteq S'$, we also have that $\mathbf{K}\,A_i \in S'$ for each $1 \leq i \leq n$ and $\mathbf{K}\,H \in T_{\mathcal{K}_G}(S')$. If $\mathbf{K}\,H \in D_{\mathcal{K}_G}(S)$, then $\mathbf{K}\,H \in M$ and $\mathsf{OB}_{\mathcal{O},S} \models H$. By monotonicity of first-order logic and since $S \subseteq S'$, we also have $\mathsf{OB}_{\mathcal{O},S'} \models H$. We conclude that $\mathbf{K}\,H \in T_{\mathcal{K}_G}(S')$. □

Since $T_{\mathcal{K}_G}$ is monotonic, it has a unique least fixpoint (by the Knaster-Tarski Theorem – see Theorem 2.2) which we denote using $T_{\mathcal{K}_G} \uparrow \omega$ in reference to the limit ordinal of natural numbers ω. It is important to note that the Knaster-Tarski Theorem in general only says that this fixpoint is reached for some ordinal that might easily be greater than ω. However, in MKNF knowledge bases, since we do not allow function symbols or infinite sets of rules, the iteration is performed over a finite knowledge base (with finitely many ground rules). As such, the iteration of $T_{\mathcal{K}_G}$ terminates for some finite ordinal below ω. The least fixpoint is obtained as follows:

$$
\begin{aligned}
T_{\mathcal{K}_G} \uparrow 0 &= \emptyset \\
T_{\mathcal{K}_G} \uparrow (n+1) &= T_{\mathcal{K}_G}(T_{\mathcal{K}_G} \uparrow n) \\
T_{\mathcal{K}_G} \uparrow \omega &= \bigcup_{i \geq 0} T_{\mathcal{K}_G} \uparrow i
\end{aligned}
$$

Example 6.12. *Consider again the hybrid MKNF knowledge base of Example 6.10. We have $T_{\mathcal{K}_G} \uparrow 0 = \emptyset$ and we compute $T_{\mathcal{K}_G} \uparrow 1 = \{\mathbf{K}\,\mathtt{P}(\mathtt{a})\}$, $T_{\mathcal{K}_G} \uparrow 2 = \{\mathbf{K}\,\mathtt{P}(\mathtt{a}), \mathbf{K}\,\mathtt{Q}(\mathtt{a})\}$, and obtain $T_{\mathcal{K}_G} \uparrow 3 = \{\mathbf{K}\,\mathtt{P}(\mathtt{a}), \mathbf{K}\,\mathtt{Q}(\mathtt{a}), \mathbf{K}\,\mathtt{r}\}$ as the fixpoint.*

Similarly to stable models of normal logic programs a fixpoint operator can be defined that performs a Gelfond-Lifschitz-like transformation [Gelfond and Lifschitz, 1988] (see also Definition 2.12) that turns hybrid MKNF knowledge bases into positive ones, and that then applies the operator $T_{\mathcal{K}_G}$ to the resulting knowledge base.

Definition 6.13. *Let $\mathcal{K}_G = (\mathcal{O}, \mathcal{P}_G)$ be a ground hybrid MKNF knowledge base and $S \subseteq \mathsf{KA}(\mathcal{K}_G)$. The* MKNF transform *$\mathcal{K}_G/S$ is defined as $\mathcal{K}_G/S = (\mathcal{O}, \mathcal{P}_G/S)$, where $\mathcal{P}_G/S$ contains all rules*

$$\mathbf{K}\, H \leftarrow \mathbf{K}\, A_1, \ldots, \mathbf{K}\, A_n$$

for which there exists a rule

$$\mathbf{K}\, H \leftarrow \mathbf{K}\, A_1, \ldots, \mathbf{K}\, A_n, \mathbf{not}\, B_1, \ldots, \mathbf{not}\, B_m$$

in $\mathcal{P}_G$ with $\mathbf{K}\, B_j \notin S$ for all $1 \leq j \leq m$.

This definition indeed resembles the transformation used to compute stable models [Gelfond and Lifschitz, 1988] of logic programs. I.e., we remove all rules that contain negated atoms contradicting the given set S, and we remove all remaining negated atoms from the other rules. Following [Gelfond and Lifschitz, 1988] (see also Definition 2.17), we define an operator that computes the least fixpoint of the resulting knowledge base.

Definition 6.14. *Let $\mathcal{K}_G = (\mathcal{O}, \mathcal{P}_G)$ be a ground hybrid MKNF knowledge base and $S \subseteq \mathsf{KA}(\mathcal{K}_G)$. We define*

$$\Gamma_{\mathcal{K}_G}(S) = T_{\mathcal{K}_G/S} \uparrow \omega.$$

Example 6.15. *Consider again the ground hybrid MKNF knowledge base $\mathcal{K}_G$ from Example 6.2.*

$$\begin{aligned} \mathtt{Q} &\sqsubseteq \mathtt{R} \\ \mathbf{K}\,\mathtt{p} &\leftarrow \mathbf{not}\,\mathtt{p} \\ \mathbf{K}\,\mathtt{Q}(\mathtt{a}) &\leftarrow \mathbf{K}\,\mathtt{R}(\mathtt{a}) \\ \mathbf{K}\,\mathtt{s} &\leftarrow \mathbf{not}\,\mathtt{Q}(\mathtt{a}) \end{aligned}$$

Given $S = \{\mathbf{K}\,\mathtt{s}\}$, we have $T_{\mathcal{K}_G/S} \uparrow \omega = \{\mathbf{K}\,\mathtt{s}, \mathbf{K}\,\mathtt{p}\}$. If we apply $\Gamma_{\mathcal{K}_G}$ to that result then we obtain again $S = \{\mathbf{K}\,\mathtt{s}\}$.

Inspired by the similarities between the definition of $\Gamma_{\mathcal{K}_G}$ and Γ in [Gelfond and Lifschitz, 1988], the correspondence of stable models for logic programs and two-valued MKNF models for knowledge bases without ontology axioms, and the results in alternating fixpoints of normal logic programs [van Gelder, 1989], one might wonder whether

iteratively applying the operator $\Gamma_{\mathcal{K}_G}$ would yield the least three-valued MKNF model and Example 6.15 seems to imply that. In fact, as shown below in Lemma 6.20, the operator $\Gamma_{\mathcal{K}_G}$ is antitonic. Thus, $\Gamma^2_{\mathcal{K}_G}$ is monotonic and guaranteed to have a least fixpoint, which can be obtained by iteratively applying $\Gamma^2_{\mathcal{K}_G}$ starting from the empty set. One may then ask whether this least fixpoint corresponds to the well-founded MKNF model. However, as shown in the example below, this is not the case, and, thus, an adaptation of alternating fixpoints to hybrid MKNF knowledge bases cannot be as straightforward.

Example 6.16. *Consider the hybrid MKNF knowledge base presented in Examples 3.18 and 4.17 for recommending CDs, and suppose now that the user wants to stall recommendations until an evaluation is available. This can be achieved, e.g., by adding the rule (6.11).*

$$\mathbf{K}\,\mathtt{LowEval}(\mathtt{x}) \leftarrow \mathbf{not}\,\mathtt{Recommend}(\mathtt{x}) \tag{6.11}$$

With this rule, together with (4.8), a CD is not recommended unless one adds explicit information that the CD has no low evaluation. To ease the reading, we recall here rule (4.8):

$$\mathbf{K}\,\mathtt{Recommend}(\mathtt{x}) \leftarrow \mathbf{K}\,\mathtt{CD}(\mathtt{x}), \mathbf{not}\,\mathtt{owns}(\mathtt{x}), \mathbf{not}\,\mathtt{LowEval}(\mathtt{x}), \mathbf{K}\,\mathtt{interesting}(\mathtt{x}). \tag{6.12}$$

In fact, if one adds, e.g.,

$$\neg\mathtt{LowEval}(\mathtt{ToTheSea}) \tag{6.13}$$

then all three-valued MKNF models contain $\mathbf{K}\,\mathtt{Recommend}(\mathtt{ToTheSea})$*. However, as shown next,* $\mathbf{K}\,\mathtt{Recommend}(\mathtt{ToTheSea})$ *is not contained in the least fixpoint of* $\Gamma^2_{\mathcal{K}_G}$*.*

To simplify the computation and presentation of this least fixpoint, we ground all the rules only with ToTheSea *(thus ignoring any other CDs), and we add explicitly that* ToTheSea *is a CD (6.14).*

$$\mathtt{CD}(\mathtt{ToTheSea}) \tag{6.14}$$

We also limit ourselves to the following set of modal atoms (using appropriate abbreviations):

$$\mathsf{KA}(\mathcal{K}_G) = \{\mathbf{K}\,\mathtt{Rec}(\mathtt{Tts}), \mathbf{K}\,\mathtt{LowEv}(\mathtt{Tts}), \mathbf{K}\,\mathtt{CD}(\mathtt{Tts}), \mathbf{K}\,\mathtt{owns}(\mathtt{Tts}), \mathbf{K}\,\mathtt{int}(\mathtt{Tts})\}$$

We start with $\mathbf{S}_0 = \emptyset$*, so we compute* $\Gamma_{\mathcal{K}_G}(\mathbf{S}_0)$ *and* $\mathbf{S}_1 = \Gamma_{\mathcal{K}_G}(\Gamma_{\mathcal{K}_G}(\mathbf{S}_0))$*:*

$$\Gamma_{\mathcal{K}_G}(\mathbf{S}_0) = \mathsf{KA}(\mathcal{K}_G)$$
$$\mathbf{S}_1 = \{\mathbf{K}\,\mathtt{CD}(\mathtt{Tts}), \mathbf{K}\,\mathtt{int}(\mathtt{Tts})\}$$

Note that, since $\mathbf{K}\,\mathtt{LowEv}(\mathtt{Tts}) \in T_{\mathcal{K}_G/\mathbf{S}_0}(\emptyset)$*, then, by (6.13),* $\mathsf{OB}_{\mathcal{O},T_{\mathcal{K}_G/\mathbf{S}_0}(\emptyset)}$ *is inconsistent. So, the subsequent application of* $D_{\mathcal{K}_G}$ *allows us to derive everything, and thus* $\Gamma_{\mathcal{K}_G}(\mathbf{S}_0) = \mathsf{KA}(\mathcal{K}_G)$*.*

We continue with $\Gamma_{\mathcal{K}_G}(\mathbf{S}_1)$ *and* $\mathbf{S}_2 = \Gamma_{\mathcal{K}_G}(\Gamma_{\mathcal{K}_G}(\mathbf{S}_1))$ *and obtain:*

$$\Gamma_{\mathcal{K}_G}(\mathbf{S}_1) = \mathsf{KA}(\mathcal{K}_G)$$
$$\mathbf{S}_2 = \{\mathbf{K}\,\mathtt{CD}(\mathtt{Tts}), \mathbf{K}\,\mathtt{int}(\mathtt{Tts})\}$$

Now, since $\mathbf{S}_1 = \mathbf{S}_2$*, the fixpoint is reached, and, indeed, does not contain* $\mathbf{K}\,\mathtt{Rec}(\mathtt{Tts})$*. This is so because, since* $\mathbf{K}\,\mathtt{LowEv}(\mathtt{Tts}) \in \Gamma_{\mathcal{K}_G}(\mathbf{S}_1)$*, rule (4.8) grounded with* Tts *is removed in* $\mathcal{P}_G/\Gamma_{\mathcal{K}_G}(\mathbf{S}_1)$*.*

Note that $\mathbf{K}\,\mathtt{LowEv}(\mathtt{Tts}) \in \Gamma_{\mathcal{K}_G}(\mathbf{S}_1)$ *because rule (6.11) is not removed in* $\mathcal{P}_G/\mathbf{S}_1$*, given that* $\mathbf{K}\,\mathtt{Rec}(\mathtt{Tts}) \notin \mathbf{S}_1$*. In an analogy with [van Gelder, 1989],* $\mathbf{K}\,\mathtt{LowEv}(\mathtt{Tts})$ *is thus either true or undefined , since it belongs to the overestimate in the alternating fixpoint. This shows that, in opposite to the three-valued MKNF semantics,* $\neg\mathtt{LowEv}(\mathtt{Tts})$ *does not imply* $\mathbf{not}\,\mathtt{LowEv}(\mathtt{Tts})$*.*[1] *In fact, for any three-valued MKNF model* (M, N) *of the restricted knowledge base, if* $\neg\mathtt{LowEv}(\mathtt{Tts})$ *holds, then, for all* $I \in M$*,* $\mathtt{LowEv}(\mathtt{Tts}) \notin \mathtt{I}$*. Thus, since* $N \subseteq M$*, we also have, for all* $I \in N$*,* $\mathtt{LowEv}(\mathtt{Tts}) \notin \mathtt{I}$*, i.e.,* $\mathbf{not}\,\mathtt{LowEv}(\mathtt{Tts})$ *should be true in any three-valued MKNF model of the knowledge base.*

One way of guaranteeing that the classical negation of some DL-atom H in the ontology imposes the truth of $\mathbf{not}\,H$ (despite the existence of rules with head $\mathbf{K}\,H$) is to change the MKNF transform defined above, so that rules with head $\mathbf{K}\,H$ are removed whenever $\neg H$ holds:

Definition 6.17. *Let* $\mathcal{K}_G = (\mathcal{O}, \mathcal{P}_G)$ *be a ground hybrid MKNF knowledge base and* $S \subseteq \mathsf{KA}(\mathcal{K}_G)$*. The* MKNF-coherent transform $\mathcal{K}_G//S$ *is defined as* $\mathcal{K}_G//S = (\mathcal{O}, \mathcal{P}_G//S)$*, where* $\mathcal{P}_G//S$ *contains all rules*

$$\mathbf{K}\,H \leftarrow \mathbf{K}\,A_1, \ldots, \mathbf{K}\,A_n$$

[1] This problem is akin to the *coherence problem* [Pereira and Alferes, 1992] in extended logic programs, where a (classical false) formula $\neg\varphi$ has to impose $\mathbf{not}\,\varphi$ explicitly.

for which there exists a rule

$$\mathbf{K}\, H \leftarrow \mathbf{K}\, A_1, \ldots, \mathbf{K}\, A_n, \mathbf{not}\, B_1, \ldots, \mathbf{not}\, B_m$$

in $\mathcal{P}_G$ *with* $\mathbf{K}\, B_j \notin S$ *for all* $1 \leq j \leq m$ *and* $\mathsf{OB}_{\mathcal{O},S} \not\models \neg H$.

Note the difference between this definition and Definition 6.13: we also remove a rule from the MKNF-coherent transform, in case the classical negation of the head is derivable from the ontology augmented by S.

Similarly to Definition 6.14, we can define an operator based on that notion of MKNF-coherent transform.

Definition 6.18. *Let* $\mathcal{K}_G = (\mathcal{O}, \mathcal{P}_G)$ *be a ground hybrid MKNF knowledge base and* $S \subseteq \mathsf{KA}(\mathcal{K}_G)$. *We define* $\Gamma'_{\mathcal{K}_G}(S) = T_{\mathcal{K}_G//S} \uparrow \omega$.

The operator $\Gamma'_{\mathcal{K}_G}$ is also antitonic (cf. Lemma 6.20), and so applying $\Gamma'_{\mathcal{K}_G}$ twice is guaranteed to have a least fixpoint. Clearly, in the case of the knowledge base of Example 6.16, this least fixpoint includes $\mathbf{K}\,\mathtt{Rec}(\mathtt{Tts})$. The reason is that the new MKNF-coherent transform does not contain any rule with head $\mathbf{K}\,\mathtt{LowEval}(\mathtt{Tts})$ once $\neg\mathtt{LowEval}(\mathtt{Tts})$ is derived, and so rule (4.8) instantiated with `Tts` is not removed at some step of the iteration. However, the next example shows that the operator $\Gamma'_{\mathcal{K}_G}$ literally hides inconsistencies from the iteration. A modal atom $\mathbf{K}\, H$ may be simply considered false, even though there is a rule with head $\mathbf{K}\, H$ such that the body is true in all three-valued MKNF models of the respective knowledge base.

Example 6.19. *Consider again only the hybrid MKNF knowledge base presented in Example 3.18 and 4.17 for recommending CDs. Now suppose that the user wants to ensure that only inexpensive CDs are to be recommended. Note that this is different from recommending CDs that have a discount. The ontology axiom (6.15) states that any expensive CD must never be recommended. In general, comparing prices requires some predicates from the numerical domain, such as concrete domains for the DL* $\mathcal{SROEL}$ *[Baader et al., 2008]. For simplicity, we assume here that this is handled internally, so we simply add a fact (6.16) saying that* `ToTheSea` *is expensive.*

$$\mathtt{Expensive} \sqsubseteq \neg\mathtt{Recommend} \tag{6.15}$$

$$\mathbf{K}\,\mathtt{Expensive}(\mathtt{ToTheSea}) \leftarrow \tag{6.16}$$

This knowledge base is clearly MKNF-inconsistent: simply note that we can conclude that `ToTheSea` *is recommended (from (4.8) instantiated by* `ToTheSea`*) and not recommended at the same time (by (6.15)–(6.16)). However, it is easy to check that the least fixpoint of applying* $\Gamma'_{\mathcal{K}_G}$ *twice does not include* $\mathbf{K}\,$`Rec(Tts)`*, simply because the MKNF-coherent transform removes the rule (4.8) instantiated with* `Tts` *once* ¬`Recommend(Tts)` *is true, even though the rule body is true.*

This example shows that $\Gamma'_{\mathcal{K}_G}$ cannot be applied always in the alternating fixpoint. Examining again the computation of Example 6.16, we may see that the application of $\Gamma'_{\mathcal{K}_G}$ would only be required when we compute overestimates of the true knowledge. In this case, it would suffice to apply $\Gamma'_{\mathcal{K}_G}$ for computing $\Gamma'_{\mathcal{K}_G}(\mathbf{S}_1)$. Then, $\mathbf{S}_2$ could be obtained by simply applying $\Gamma_{\mathcal{K}_G}$ to the previous result, thus computing $\mathbf{S}_2$ now by $\Gamma_{\mathcal{K}_G}(\Gamma'_{\mathcal{K}_G}(\mathbf{S}_1))$. At the same time, in case of Example 6.19, the partial usage of $\Gamma_{\mathcal{K}_G}$ would ensure that `Recommend(Tts)` is kept in one part of the iteration so that inconsistencies may still be detectable.

This suggests that the computation of the well-founded MKNF model could be obtained by alternating the application of the operators $\Gamma'_{\mathcal{K}_G}$ and $\Gamma_{\mathcal{K}_G}$. In fact, this interaction of the two operators yields the well-founded MKNF model. But before we formally define this interaction and prove its correspondence to the well-founded MKNF model, we show that both operators are indeed antitonic.

Lemma 6.20. *If* $\mathcal{K}_G$ *is a ground hybrid MKNF knowledge base and* $S \subseteq S' \subseteq \mathsf{KA}(\mathcal{K}_G)$*, then* $\Gamma_{\mathcal{K}_G}(S') \subseteq \Gamma_{\mathcal{K}_G}(S)$ *and* $\Gamma'_{\mathcal{K}_G}(S') \subseteq \Gamma'_{\mathcal{K}_G}(S)$*.*

Proof. We show the argument for $\Gamma_{\mathcal{K}_G}$. The proof for $\Gamma'_{\mathcal{K}_G}$ is identical.

By Definition 6.14, we have to show that $T_{\mathcal{K}_G/S'} \uparrow \omega \subseteq T_{\mathcal{K}_G/S} \uparrow \omega$. We prove by induction on n that $T_{\mathcal{K}_G/S'} \uparrow n \subseteq T_{\mathcal{K}_G/S} \uparrow n$ holds. The base case for $n = 0$ is trivial since $\emptyset \subseteq \emptyset$. Assume that $T_{\mathcal{K}_G/S'} \uparrow n \subseteq T_{\mathcal{K}_G/S} \uparrow n$ holds and consider $\mathbf{K}\,H \in T_{\mathcal{K}_G/S'} \uparrow (n+1)$. Then $\mathbf{K}\,H \in T_{\mathcal{K}_G/S'}(T_{\mathcal{K}_G/S'} \uparrow n)$ and there are two cases to consider. First, $\mathcal{K}_G/S'$ contains a rule of the form $\mathbf{K}\,H \leftarrow \mathbf{K}\,A_1, \ldots, \mathbf{K}\,A_n$ such that $\mathbf{K}\,A_i \in T_{\mathcal{K}_G/S'} \uparrow n$ for each $1 \leq i \leq n$. Since $S \subseteq S'$, we also have $\mathbf{K}\,H \leftarrow \mathbf{K}\,A_1, \ldots, \mathbf{K}\,A_n$ in $\mathcal{K}_G/S$ and, by the induction hypothesis, $\mathbf{K}\,A_i \in T_{\mathcal{K}_G/S} \uparrow n$ for each $1 \leq i \leq n$. Hence, $\mathbf{K}\,H \in T_{\mathcal{K}_G/S} \uparrow (n+1)$. Alternatively, $\mathbf{K}\,H$ is a consequence obtained from $D_{\mathcal{K}_G/S'}(T_{\mathcal{K}_G/S'} \uparrow n)$. By the induction hypothesis, $T_{\mathcal{K}_G/S'} \uparrow n \subseteq T_{\mathcal{K}_G/S} \uparrow n$ holds, and we conclude from the monotonicity of first-order logic that $\mathbf{K}\,H \in D_{\mathcal{K}_G/S}(T_{\mathcal{K}_G/S} \uparrow n)$. □

Since both operators are antitonic, we can define an alternating iteration for the two operators as motivated above:

Definition 6.21. *Let $\mathcal{K}_G$ be a ground hybrid MKNF knowledge base. We define two sequences $\mathbf{P}_i$ and $\mathbf{N}_i$ as follows.*

$$\begin{aligned} \mathbf{P}_0 &= \emptyset & \mathbf{N}_0 &= \mathsf{KA}(\mathcal{K}_G) \\ \mathbf{P}_{n+1} &= \Gamma_{\mathcal{K}_G}(\mathbf{N}_n) & \mathbf{N}_{n+1} &= \Gamma'_{\mathcal{K}_G}(\mathbf{P}_n) \\ \mathbf{P}_\omega &= \bigcup \mathbf{P}_i & \mathbf{N}_\omega &= \bigcap \mathbf{N}_i \end{aligned}$$

The sequence of $\mathbf{P}_i$ is intended to compute modal atoms that are true, while the sequence $\mathbf{N}_i$ computes modal atoms that are not false. The former is an increasing sequence, while the latter is decreasing:

Lemma 6.22. *Let $\mathcal{K}_G$ be a ground hybrid MKNF knowledge base. Then $\mathbf{P}_\alpha \subseteq \mathbf{P}_\beta$ and $\mathbf{N}_\beta \subseteq \mathbf{N}_\alpha$ for all ordinals α, β with $\alpha \leq \beta \leq \omega$.*

Proof. Whenever $\alpha = \beta$, the statement holds automatically. It thus suffices to consider $\alpha < \omega$ and to show via induction over α that the statement holds. If β is a successor ordinal, then it is sufficient to show the property for $\beta = \alpha + 1$, all the other successor cases follow by transitivity of $\subseteq$.

If $\alpha = 0$, then $\mathbf{P}_0 = \emptyset$ and $\mathbf{P}_0 \subseteq \mathbf{P}_\beta$ holds for any β. Equivalently, $\mathbf{N}_0 = \mathsf{KA}(\mathcal{K}_G)$, thus $\mathbf{N}_\beta \subseteq \mathbf{N}_0$ also holds for any β.

Suppose the property holds for all $\alpha \leq n$. We must show that $\mathbf{P}_{n+1} \subseteq \mathbf{P}_{n+2}$ and $\mathbf{N}_{n+2} \subseteq \mathbf{N}_{n+1}$. We have $\mathbf{P}_{n+1} = \Gamma(\mathbf{N}_n)$ and $\mathbf{P}_{n+2} = \Gamma(\mathbf{N}_{n+1})$. Since $\mathbf{N}_{n+1} \subseteq \mathbf{N}_n$ by the induction hypothesis, $\mathbf{P}_{n+1} \subseteq \mathbf{P}_{n+2}$ holds in virtue of the anti-monotonicity of Γ'. Likewise, we know that $\mathbf{N}_{n+1} = \Gamma'(\mathbf{P}_n)$ and $\mathbf{N}_{n+2} = \Gamma'(\mathbf{P}_{n+1})$. Since $\mathbf{P}_n \subseteq \mathbf{P}_{n+1}$ by the induction hypothesis, we obtain by anti-monotonicity of Γ' that $\mathbf{N}_{n+2} \subseteq \mathbf{N}_{n+1}$.

The only case left is the one where $\beta = \omega$. But this case holds by definition. □

Like the iteration of $T_\mathcal{K}$, and for the very same reasons, these iterations are finite and reach a fixpoint before ω – in the case of $\mathbf{P}_i$ a least fixpoint, and in the case of $\mathbf{N}_i$ a greatest fixpoint:

Proposition 6.23. *Let $\mathcal{K}_G$ be a ground hybrid MKNF knowledge base. Then $\mathbf{P}_\omega$ is the least fixpoint of the sequence of $\mathbf{P}_i$ and $\mathbf{N}_\omega$ is the greatest fixpoint of the sequence of $\mathbf{P}_i$.*

Proof. We show the argument for $\mathbf{P}_\omega$. The argument for $\mathbf{N}_\omega$ is analogous.

We define an operator $\Phi(S) = \Gamma_{\mathcal{K}_G}(\Gamma'_{\mathcal{K}_G}(S))$ on subsets S of $\mathsf{KA}(\mathcal{K}_G)$, iterated as usual. It is easy to see that $\Phi \uparrow i = \mathbf{P}_{2i}$ and, thus, that Φ is monotonic. By the Knaster-Tarski Theorem 2.2, we conclude that $\mathbf{P}_\omega$ is equal to the least fixpoint of the sequence of $\mathbf{P}_i$. □

This proposition also allows us to show that the least fixpoint can be directly computed from the greatest one and vice versa.

Proposition 6.24. *Let $\mathcal{K}_G$ be a ground hybrid MKNF knowledge base. Then $\mathbf{P}_\omega = \Gamma_{\mathcal{K}_G}(\mathbf{N}_\omega)$ and $\mathbf{N}_\omega = \Gamma'_{\mathcal{K}_G}(\mathbf{P}_\omega)$.*

Proof. We show the case of $\mathbf{N}_\omega = \Gamma'(\mathbf{P}_\omega)$; the other case proceeds identically. By Proposition 6.23, we know that $\mathbf{P}_\omega$ is the least fixpoint of the sequence of $\mathbf{P}_i$. Since the ground knowledge base is finite, there is an n such that $\mathbf{P}_n = \mathbf{P}_\omega$, and so $\mathbf{P}_n = \mathbf{P}_m$ for any $m \geq n$. Subsequently, we have $\mathbf{N}_{n+1} = \mathbf{N}_m$ for any m with $m \geq n+1$, i.e., $\mathbf{N}_{n+1} = \Gamma'_{\mathcal{K}_G}(\mathbf{P}_\omega)$ is a fixpoint of the sequence $\mathbf{N}_i$. Assume that $\mathbf{N}_{n+1}$ is not the greatest fixpoint. Then there is an $\mathbf{N}_l$, $l < n+1$, with $\mathbf{N}_l = \mathbf{N}_{l+2}$ and $\mathbf{N}_l \supset \mathbf{N}_{n+1}$. Then $\mathbf{P}_{l+1}$ also equals a fixpoint in the sequence $\mathbf{P}_i$ with $\mathbf{P}_{l+1}$ being necessarily smaller than $\mathbf{P}_n$. This contradicts the initial assumption that $\mathbf{P}_n$ is the least fixpoint and finishes the proof. □

Thus, we can either compute the two sequences $\mathbf{P}_i$ and $\mathbf{N}_i$ in parallel until we reach an n such that $\mathbf{P}_n = \mathbf{P}_{n+1}$ and $\mathbf{N}_n = \mathbf{N}_{n+1}$ or we compute just one of the two fixpoints in the manner sketched in the proof of Proposition 6.23 (alternating between $\Gamma_{\mathcal{K}_G}$ and $\Gamma'_{\mathcal{K}_G}$) and let the other one follow by one application of either $\Gamma_{\mathcal{K}_G}$ or $\Gamma'_{\mathcal{K}_G}$.

The two fixpoints can be used to define the well-founded partition which is, as we show in Section 6.4, the partition inducing the well-founded MKNF model.

Definition 6.25. *The* well-founded partition *of an MKNF-consistent ground hybrid MKNF knowledge base $\mathcal{K}_G = (\mathcal{O}, \mathcal{P}_G)$ is defined by:*

$$(T_W, F_W) = (\mathbf{P}_\omega, \mathsf{KA}(\mathcal{K}_G) \setminus \mathbf{N}_\omega)$$

Note that we restrict the definition to MKNF-consistent hybrid MKNF knowledge bases. This is reasonable since in many cases the pair (T_W, F_W) obtained for an MKNF-inconsistent knowledge base would not satisfy the conditions imposed in the definition of a partition (cf. Definition 6.1). Therefore, in Section 6.4, we show that all **K**-atoms derived in $\mathbf{P}_\omega$, $\mathsf{KA}(\mathcal{K}_G) \setminus \mathbf{N}_\omega$ respectively, are true, false respectively, in all

three-valued MKNF models of $\mathcal{K}_G$ (see Proposition 6.38), including the special case, in which $\mathcal{K}_G$ is MKNF-inconsistent. This can be used to present necessary and sufficient conditions to check for MKNF-consistency (see Theorem 6.39), which are based on two comparisons, each of which compares a further iteration of the operators $\Gamma_{\mathcal{K}_G}$ and $\Gamma'_{\mathcal{K}_G}$ w.r.t. one of the fixpoints. Given an established check for MKNF-consistency, we can show that the well-founded partition is in fact a partial partition (see Proposition 6.42), if the considered knowledge base $\mathcal{K}_G$ is consistent. In this case, we can also show that a corresponding MKNF interpretation pair exists that satisfies $\mathcal{K}_G$ (see Theorem 6.43), and that this interpretation pair is a three-valued MKNF model of $\mathcal{K}_G$ (see Theorem 6.44). This allows us to conclude that this specific MKNF interpretation pair corresponding to the well-founded partition is the well-founded MKNF model (see Theorem 6.45). We can then show that, given $\mathcal{K}_G$ with empty $\mathcal{O}$, the well-founded partition and the well-founded model for logic programs coincide (see Theorem 6.47) and finish with the results for data complexity (see Theorem 6.48). But before we come to that (in Section 6.4), we illustrate the alternating fixpoint construction in the two motivating examples (Examples 6.16, 6.19) presented before.

Example 6.26. *Consider the hybrid MKNF knowledge base presented in Example 6.16, with the same limitation on the set of modal atoms.*

For similarity with the computation presented in Example 6.16, we compute the fixpoints as sketched in the proof of Proposition 6.23. We start with $\mathbf{P}_0 = \emptyset$ *and compute* $\mathbf{N}_1 = \Gamma'_{\mathcal{K}_G}(\mathbf{P}_0)$ *and* $\mathbf{P}_2 = \Gamma_{\mathcal{K}_G}(\mathbf{N}_1)$*:*

$$\mathbf{N}_1 = \{\mathbf{K}\,\mathtt{CD}(\mathtt{Tts}), \mathbf{K}\,\mathtt{int}(\mathtt{Tts}), \mathbf{K}\,\mathtt{Rec}(\mathtt{Tts})\}$$
$$\mathbf{P}_2 = \{\mathbf{K}\,\mathtt{CD}(\mathtt{Tts}), \mathbf{K}\,\mathtt{int}(\mathtt{Tts}), \mathbf{K}\,\mathtt{Rec}(\mathtt{Tts})\}$$

It is easy to check that $\mathbf{P}_2$ *is already the least fixpoint. Note the difference to the iteration in Example 6.16. Now,* $\mathbf{K}\,\mathtt{LowEv}(\mathtt{Tts})$ *does not occur in* $\mathbf{N}_1$*, since* $\mathsf{OB}_{\mathcal{O},P_0} \models \neg\mathtt{LowEv}(\mathtt{Tts})$*. So rule (6.11) instantiated with* Tts *is removed in the MKNF coherent-transform, and thus* $\mathbf{K}\,\mathtt{LowEv}(\mathtt{Tts}) \notin \Gamma'_{\mathcal{K}_G}(\mathbf{P}_0)$*. As a consequence, we obtain* $\mathbf{K}\,\mathtt{Rec}(\mathtt{Tts})$ *in* $\mathbf{P}_2$*. We can compute the greatest fixpoint* $\mathbf{N}_\omega = \Gamma'_{\mathcal{K}_G}(\mathbf{P}_\omega)$*, and we obtain that* $\mathbf{N}_\omega$ *equals* $\mathbf{N}_1$*. Note that if axiom (6.13) is omitted, then both* $\mathbf{K}\,\mathtt{LowEv}(\mathtt{Tts})$ *and* $\mathbf{K}\,\mathtt{Rec}(\mathtt{Tts})$ *remain undefined. Thus, operator* $\Gamma'_{\mathcal{K}_G}$*, in combination with (6.13), shows how the formula* $\neg\mathtt{LowEv}(\mathtt{Tts})$ *imposes that* **not** $\mathtt{LowEv}(\mathtt{Tts})$ *holds, ensuring in this example the derivability of* $\mathbf{K}\,\mathtt{Rec}(\mathtt{Tts})$*.*

The knowledge base in Example 6.19 is MKNF-inconsistent. Thus, there cannot be a well-founded partition as in Definition 6.25. Nevertheless, we present the computation of the alternating fixpoint, in order to show the difference to the computation in Example 6.19, and to hint on how to detect inconsistencies, a topic that is detailed in Section 6.4.

Example 6.27. *Consider the ground hybrid MKNF knowledge base $\mathcal{K}_G$ presented in Example 6.19, where all rules are only grounded with* `Tts`*, and the subsequent restricted set of modal atoms is used:*

$$\begin{aligned}\mathsf{KA}(\mathcal{K}_G) = \{&\mathbf{K}\,\mathtt{Rec}(\mathtt{Tts}), \mathbf{K}\,\mathtt{CD}(\mathtt{Tts}), \mathbf{K}\,\mathtt{LowEv}(\mathtt{Tts}), \mathbf{K}\,\mathtt{owns}(\mathtt{Tts}),\\ &\mathbf{K}\,\mathtt{int}(\mathtt{Tts}), \mathbf{K}\,\mathtt{Exp}(\mathtt{Tts})\}\end{aligned}$$

To further simplify the presentation, we only consider rule (4.8), axioms (6.14)–(6.16), and we simplify (4.9) to the fact (6.21). To ease the reading we repeat here the complete knowledge base obtained after all simplifications:

$$\mathtt{Exp} \sqsubseteq \neg\mathtt{Rec} \tag{6.17}$$

$$\mathtt{CD}(\mathtt{Tts}) \tag{6.18}$$

$$\mathbf{K}\,\mathtt{Exp}(\mathtt{Tts}) \leftarrow \tag{6.19}$$

$$\mathbf{K}\,\mathtt{Rec}(\mathtt{Tts}) \leftarrow \mathbf{K}\,\mathtt{CD}(\mathtt{Tts}), \mathbf{not}\,\mathtt{owns}(\mathtt{Tts}), \mathbf{not}\,\mathtt{LowEval}(\mathtt{Tts}), \mathbf{K}\,\mathtt{int}(\mathtt{Tts}). \tag{6.20}$$

$$\mathbf{K}\,\mathtt{int}(\mathtt{Tts}) \leftarrow \tag{6.21}$$

For computing the two fixpoints, we start with $\mathbf{P}_0 = \emptyset$ and $\mathbf{N}_0 = \mathsf{KA}(\mathcal{K}_G)$. We continue with $\mathbf{P}_1 = \Gamma_{\mathcal{K}_G}(\mathbf{N}_0)$ and $\mathbf{N}_1 = \Gamma'_{\mathcal{K}_G}(\mathbf{P}_0)$:

$$\begin{aligned}\mathbf{P}_1 &= \{\mathbf{K}\,\mathtt{CD}(\mathtt{Tts}), \mathbf{K}\,\mathtt{int}(\mathtt{Tts}), \mathbf{K}\,\mathtt{Exp}(\mathtt{Tts})\}\\ \mathbf{N}_1 &= \mathsf{KA}(\mathcal{K}_G)\end{aligned}$$

Note that once $\mathbf{K}\,\mathtt{Rec}(\mathtt{Tts})$ is derived in the computation of $\mathbf{N}_1$ and added to the set S of derived knowledge of $T_{\mathcal{K}_G/\emptyset}$, then $D_{\mathcal{K}_G/\emptyset}$ allows us to derive everything, simply because $\mathsf{OB}_{\mathcal{O},S}$ with $\{\mathbf{K}\,\mathtt{Exp}(\mathtt{Tts}), \mathbf{K}\,\mathtt{Rec}(\mathtt{Tts})\} \subseteq \mathtt{S}$ is inconsistent.

We continue with $\mathbf{P}_2 = \Gamma_{\mathcal{K}_G}(\mathbf{N}_1)$ and $\mathbf{N}_2 = \Gamma'_{\mathcal{K}_G}(\mathbf{P}_1)$, and obtain:

$$\begin{aligned}\mathbf{P}_2 &= \{\mathbf{K}\,\mathtt{CD}(\mathtt{Tts}), \mathbf{K}\,\mathtt{int}(\mathtt{Tts}), \mathbf{K}\,\mathtt{Exp}(\mathtt{Tts})\}\\ \mathbf{N}_2 &= \{\mathbf{K}\,\mathtt{CD}(\mathtt{Tts}), \mathbf{K}\,\mathtt{int}(\mathtt{Tts}), \mathbf{K}\,\mathtt{Exp}(\mathtt{Tts})\}\end{aligned}$$

Since $\mathsf{OB}_{\mathcal{O},P_1} \models \neg\mathtt{Rec}(\mathtt{Tts})$*, the rule (6.20) no longer appears in the transform used for computing* $\mathbf{N}_2$*, and the explosive behavior of* $D_{\mathcal{K}_G/\emptyset}$ *disappears as well. As a consequence, in the next iteration we obtain* $\mathbf{K}\,\mathtt{Rec}(\mathtt{Tts}) \in \mathbf{P}_3$*, which again yields the explosive inconsistency and the derivation of* $\mathsf{KA}(\mathcal{K}_G)$*.*

$$\mathbf{P}_3 = \mathsf{KA}(\mathcal{K}_G)$$
$$\mathbf{N}_3 = \{\mathbf{K}\,\mathtt{CD}(\mathtt{Tts}), \mathbf{K}\,\mathtt{int}(\mathtt{Tts}), \mathbf{K}\,\mathtt{Exp}(\mathtt{Tts})\}$$

It is easy to check that these are the fixpoints. We obtain that $\mathbf{K}\,\mathtt{Rec}(\mathtt{Tts}) \in \mathbf{P}_3$ *but* $\mathbf{K}\,\mathtt{Rec}(\mathtt{Tts}) \notin \mathbf{N}_3$*. Intuitively, this means that* $\mathbf{K}\,\mathtt{Rec}(\mathtt{Tts})$ *is true and false at the same time, something that is already a clear indication for the inconsistency of the considered knowledge base.*

6.3 An Alternative Characterization based on Unfounded Sets

As outlined at the end of the previous section, we are going to show several important properties of our newly defined three-valued MKNF semantics, the unique well-founded MKNF model, and the just defined procedure. However, the alternating fixpoint construction is sometimes difficult to use in proofs since it is not really obvious in each situation why a certain modal atom is derived to be false. For logic programs, the notion of unfounded sets (Definition 2.25 in Chapter 2) provides a much more declarative way than the alternating fixpoint. This is why we also provide an alternative definition of the iteration based on unfounded sets.

For that purpose we need to define a notion of dependency that captures more precisely the derivations from $\mathsf{OB}_{\mathcal{O},S}$, for some S, by the operator $D_{\mathcal{K}_G}$.

Definition 6.28. *Let* $\mathcal{K}_G = (\mathcal{O}, \mathcal{P}_G)$ *be a ground hybrid MKNF knowledge base,* $\mathbf{K}\,H$ *a modal* $\mathbf{K}$*-atom with* $\mathbf{K}\,H \in \mathsf{KA}(\mathcal{K}_G)$*, and* S *a (possibly empty) set of modal* $\mathbf{K}$*-atoms with* $S \subseteq \mathsf{KA}(\mathcal{K}_G)$*. We say that* $\mathbf{K}\,H$ depends *on* S *if and only if*

(i) $\mathsf{OB}_{\mathcal{O},S} \models H$ *and*

(ii) there is no S' *with* $S' \subset S$ *such that* $\mathsf{OB}_{\mathcal{O},S'} \models H$*.*

Intuitively, S is a minimal set that, in combination with $\mathcal{O}$, allows us to derive $\mathbf{K}\,H$. Note that there may exist several such minimal sets.

Example 6.29. *Consider the hybrid MKNF knowledge base $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ with $\mathcal{O}$ given as follows.*

$$\mathtt{P} \sqcap \mathtt{Q} \sqsubseteq \mathtt{R} \tag{6.22}$$

$$\mathtt{R} \sqcup \mathtt{S} \sqsubseteq \mathtt{T} \tag{6.23}$$

$$\neg \mathtt{R}(\mathtt{b}) \tag{6.24}$$

Now, let $\mathsf{KA}(\mathcal{K}_G) = \{\mathbf{K}\,\mathtt{P}(\mathtt{a}), \mathbf{K}\,\mathtt{Q}(\mathtt{a}), \mathbf{K}\,\mathtt{R}(\mathtt{a}), \mathbf{K}\,\mathtt{R}(\mathtt{b}), \mathbf{K}\,\mathtt{S}(\mathtt{a}), \mathbf{K}\,\mathtt{T}(\mathtt{a})\}$*. Clearly,* $\mathbf{K}\,\mathtt{T}(\mathtt{a})$ *depends on* $\{\mathbf{K}\,\mathtt{S}(\mathtt{a})\}$*, but also on* $\{\mathbf{K}\,\mathtt{R}(\mathtt{a})\}$*. Of course, it also depends on* $\{\mathbf{K}\,\mathtt{P}(\mathtt{a}), \mathbf{K}\,\mathtt{Q}(\mathtt{a})\}$*. We can see that this notion captures which minimal set of atoms has to be added to derive a certain* $\mathbf{K}$*-atom. In the computation in the previous section, we do not care about minimality, since also the union of the three sets on which* $\mathbf{K}\,\mathtt{T}(\mathtt{a})$ *depends allow us to derive* $\mathbf{K}\,\mathtt{T}(\mathtt{a})$ *together with* $\mathcal{O}$*. However, in the forthcoming alternative definition based on unfounded sets we exactly want to pinpoint which atoms have to be false to ensure that some modal atom can never be derived. Note that* $\mathbf{K}\,\mathtt{T}(\mathtt{a})$ *also depends on* $\{\mathbf{K}\,\mathtt{R}(\mathtt{b})\}$*, i.e., we may add some atoms such that the combined* $\mathcal{O}$ *is inconsistent. This is usually not intended, so we have to restrict the usage of the notion of dependency appropriately.*

We can use this notion of dependency to introduce unfounded sets for hybrid MKNF knowledge bases.

Definition 6.30. *Let* $\mathcal{K}_G$ *be a ground hybrid MKNF knowledge base and* (T, F) *a pair of sets of* $\mathbf{K}$*-atoms with* $T, F \subseteq \mathsf{KA}(\mathcal{K}_G)$*. We say that* $U \subseteq \mathsf{KA}(\mathcal{K}_G)$ *is an* unfounded set *(of* $\mathcal{K}_G$*)* with respect to (T, F) *if, for each* $\mathbf{K}$*-atom* $\mathbf{K}\,H \in U$*, the following conditions are satisfied:*

(Ui) for each rule $\mathbf{K}\,H \leftarrow \mathcal{B}$ *in* $\mathcal{P}_G$ *at least one of the following holds.*

- *(Uia) Some* $\mathbf{K}$*-atom* $\mathbf{K}\,A$ *appears in* $\mathcal{B}$ *and in* $U \cup F$*.*
- *(Uib) Some* **not***-atom* **not** B *appears in* $\mathcal{B}$ *and in* T*.*
- *(Uic)* $\mathsf{OB}_{\mathcal{O},T} \models \neg H$

(Uii) for each (possibly empty) S *on which* $\mathbf{K}\,H$ *depends, with* $S \subseteq \mathsf{KA}(\mathcal{K}_G)$ *and* $\mathsf{OB}_{\mathcal{O},S}$ *consistent, there is at least one modal* $\mathbf{K}$*-atom* $\mathbf{K}\,A$ *such that* $\mathsf{OB}_{\mathcal{O},S\setminus\{\mathbf{K}\,A\}} \not\models H$ *and* $\mathbf{K}\,A$ *in* $U \cup F$*.*

The union of all unfounded sets of $\mathcal{K}_G$ *w.r.t.* (T, F) *is called the* greatest unfounded set *of* $\mathcal{K}_G$ *w.r.t.* (T, F)*.*

The cases of (Uia) and (Uib) are variations of the cases (Ui) and (Uii) of Definition 2.25 dealing with rules that contain some (other) false modal atom. The case of (Uic) corresponds to the removal of rules performed for the MKNF-coherent transform due to the derivability of the classical negation of the rule head (cf. Definition 6.17). The remaining condition (Uii) checks that the each $\mathbf{K}$-atom in U is no longer derivable from $\mathcal{O}$.

Example 6.31. *Consider again the KB $\mathcal{K}_G$ from Example 6.29. Assume that it also contains the following MKNF rules.*

$$\mathbf{K}\,\mathtt{P}(\mathtt{a}) \leftarrow \tag{6.25}$$

$$\mathbf{K}\,\mathtt{Q}(\mathtt{a}) \leftarrow \mathbf{K}\,\mathtt{R}(\mathtt{a}) \tag{6.26}$$

$$\mathbf{K}\,\mathtt{R}(\mathtt{a}) \leftarrow \mathbf{K}\,\mathtt{S}(\mathtt{a}) \tag{6.27}$$

$$\mathbf{K}\,\mathtt{S}(\mathtt{a}) \leftarrow \mathbf{K}\,\mathtt{T}(\mathtt{a}) \tag{6.28}$$

$$\mathbf{K}\,\mathtt{T}(\mathtt{a}) \leftarrow \mathbf{not}\,\mathtt{P}(\mathtt{a}) \tag{6.29}$$

$$\mathbf{K}\,\mathtt{R}(\mathtt{b}) \leftarrow \mathbf{not}\,\mathtt{R}(\mathtt{b}) \tag{6.30}$$

For $(\{\mathbf{K}\,\mathtt{P}(\mathtt{a})\},\{\mathbf{K}\,\mathtt{R}(\mathtt{b})\})$, we obtain that $U = \{\mathbf{K}\,\mathtt{Q}(\mathtt{a}), \mathbf{K}\,\mathtt{R}(\mathtt{a}), \mathbf{K}\,\mathtt{S}(\mathtt{a}), \mathbf{K}\,\mathtt{T}(\mathtt{a})\}$. In rule 6.29, the body is false by (Uib) and all the other rules whose heads appear in U have a false body by (Uia). Consequently, none of the elements is derivable from $\mathcal{O}$ either, due to (Uii). Note that $\mathtt{R}(\mathtt{b})$ can be obtained as an unfounded set w.r.t. $(\emptyset,\emptyset)$.

Similar to logic programs, we can define the computation of the well-founded MKNF model based on unfounded sets. For that purpose, we only have to adapt the operators from Definition 6.9 to rules that are not necessarily positive (cf. Definition 2.26).

Definition 6.32. *Let $\mathcal{K}_G = (\mathcal{O}, \mathcal{P}_G)$ be a ground hybrid MKNF knowledge base and (T, F) a pair of sets of $\mathbf{K}$-atoms with $T, F \in \mathsf{KA}(\mathcal{K}_G)$. The operators $R^u_{\mathcal{K}_G}$, $D^u_{\mathcal{K}_G}$, and $T^u_{\mathcal{K}_G}$ are defined on (T, F):*

$$\begin{aligned} R^u_{\mathcal{K}_G}(T,F) = \{&\mathbf{K}\,H \mid \mathcal{P}_G \text{ contains a rule of the form } H \leftarrow \mathcal{B}^+ \wedge \mathcal{B}^- \text{ such that,} \\ &\text{for all } \mathbf{K}\,A_i \in \mathcal{B}^+, \mathbf{K}\,A_i \in T \text{ and, for all } \mathbf{not}\,B_j \in \mathcal{B}^-, \mathbf{K}\,B_j \in F\} \end{aligned}$$

$$D^u_{\mathcal{K}_G}(T,F) = \{\mathbf{K}\,\xi \mid \mathbf{K}\,\xi \in \mathsf{KA}(\mathcal{K}_G) \text{ and } \mathsf{OB}_{\mathcal{O},T} \models \xi\}$$

$$T^u_{\mathcal{K}_G}(T,F) = R^u_{\mathcal{K}_G}(T,F) \cup D^u_{\mathcal{K}_G}(T,F)$$

The only important differences to Definition 6.9 are that all operators are defined for pairs of **K**-atoms and that $R^u_{\mathcal{K}_G}$ is taking **not**-atoms into account. Together with the previously defined notion of greatest unfounded set, we obtain the complete operator that derives information from hybrid MKNF knowledge bases.

Definition 6.33. *Let $\mathcal{K}_G = (\mathcal{O}, \mathcal{P}_G)$ be a ground hybrid MKNF knowledge base and $U_{\mathcal{K}_G}(T, F)$ the greatest unfounded set (of $\mathcal{K}_G$) with respect to (T, F). Then for all pairs of sets of* **K***-atoms (T, F) with $T, F \in \mathsf{KA}(\mathcal{K}_G)$:*

$$W_{\mathcal{K}_G}(T, F) = (T^u_{\mathcal{K}_G}(T, F), U_{\mathcal{K}_G}(T, F)).$$

We define the partial order $\subseteq$ on pairs of **K***-atoms (T_1, F_1) and (T_2, F_2):*

$$(T_1, F_1) \subseteq (T_2, F_2) \text{ iff } T_1 \subseteq T_2 \text{ and } F_1 \subseteq F_2.$$

Using this order, we can show that this operator is monotonic on T and F.

Proposition 6.34. *Let $\mathcal{K}_G = (\mathcal{O}, \mathcal{P}_G)$ be a ground hybrid MKNF knowledge base and (T_1, F_1) and (T_2, F_2) pairs of sets of* **K***-atoms (T, F) with $T, F \in \mathsf{KA}(\mathcal{K}_G)$ and $(T_1, F_1) \subseteq (T_2, F_2)$. Then $W_{\mathcal{K}_G}$ is monotonic, i.e., $W_{\mathcal{K}_G}(T_1, F_1) \subseteq W_{\mathcal{K}_G}(T_2, F_2)$.*

Proof. Consider (T_1, F_1) and (T_2, F_2) with $(T_1, F_1) \subseteq (T_2, F_2)$. We have to show that $T^u_{\mathcal{K}_G}(T_1, F_1) \subseteq T^u_{\mathcal{K}_G}(T_2, F_2)$ and $U_{\mathcal{K}_G}(T_1, F_1) \subseteq U_{\mathcal{K}_G}(T_2, F_2)$ holds.

If $\mathbf{K}H \in T^u_{\mathcal{K}_G}(T_1, F_1)$, then $\mathbf{K}H \in R^u_{\mathcal{K}_G}(T_1, F_1) \cup D^u_{\mathcal{K}_G}(T_1, F_1)$. If $\mathbf{K}H \in R^u_{\mathcal{K}_G}(T_1, F_1)$, then $\mathcal{P}_G$ contains a rule of the form $H \leftarrow \mathcal{B}^+ \wedge \mathcal{B}^-$ such that, for all $\mathbf{K}A_i \in \mathcal{B}^+$, $\mathbf{K}A_i \in T_1$ and, for all $\mathbf{not}\, B_j \in \mathcal{B}^-$, $\mathbf{K}B_j \in F_1$. It follows immediately that $\mathbf{K}H \in R^u_{\mathcal{K}_G}(T_2, F_2)$. If $\mathbf{K}H \in D^u_{\mathcal{K}_G}(T_1, F_1)$, then $\mathsf{OB}_{\mathcal{O},T_1} \models H$. We derive $\mathbf{K}H \in D^u_{\mathcal{K}_G}(T_2, F_2)$, and, therefore, $\mathbf{K}H \in T^u_{\mathcal{K}_G}(T_2, F_2)$.

Now, consider $\mathbf{K}H \in U_{\mathcal{K}_G}(T_1, F_1)$. By Definition 6.30, we derive $\mathbf{K}H \in U_{\mathcal{K}_G}(T_2, F_2)$. □

Thus, by Theorem 2.2, $W_{\mathcal{K}_G}$ has a least fixed point, which can be computed as usual by $W_{\mathcal{K}_G} \uparrow 0 = (\emptyset, \emptyset)$, $W_{\mathcal{K}_G} \uparrow (n+1) = W_{\mathcal{K}_G}(W_{\mathcal{K}_G} \uparrow n)$, and $W_{\mathcal{K}_G} \uparrow \alpha = \bigcup_{i<\alpha} W_{\mathcal{K}_G} \uparrow i$ for limit ordinals α.

Now, we can show that this least fixed point coincides with the two fixpoints of the sequences $\mathbf{P}_i$ and $\mathbf{N}_i$ (see Definition 6.21).

Theorem 6.35. *Let $\mathcal{K}_G = (\mathcal{O}, \mathcal{P}_G)$ be a ground hybrid MKNF knowledge base and (T, F) the least fixed point of $W_{\mathcal{K}_G}$. Then, (T, F) coincides with $\mathbf{P}_\omega$ and $\mathbf{N}_\omega$ as follows:*

$$(T, F) = (\mathbf{P}_\omega, \mathsf{KA}(\mathcal{K}_G) \setminus \mathbf{N}_\omega).$$

Proof. We show that $(\mathbf{P}_\omega, \mathsf{KA}(\mathcal{K}_G) \setminus \mathbf{N}_\omega)$ is a fixpoint of $W_{\mathcal{K}_G}$. This proves that $T \subseteq \mathbf{P}_\omega$ and $F \subseteq \mathsf{KA}(\mathcal{K}_G) \setminus \mathbf{N}_\omega$ since (T, F) is the least fixpoint of $W_{\mathcal{K}_G}$.

Let $\mathbf{K}\, H \in \mathbf{P}_\omega$. Then $\mathbf{K}\, H \in \Gamma_{\mathcal{K}_G}(\mathbf{N}_\omega)$, i.e., $\mathbf{K}\, H \in T_{\mathcal{K}_G/\mathbf{N}_\omega} \uparrow \omega$. Thus, there either is a rule of the form $H \leftarrow \mathcal{B}^+ \wedge \mathcal{B}^-$ in $\mathcal{P}_G$ such that, for all $\mathbf{K}\, A_i \in \mathcal{B}^+$, $\mathbf{K}\, A_i \in \mathbf{P}_\omega$, and, for all $\mathbf{not}\, B_j \in \mathcal{B}^-$, $\mathbf{K}\, B_j \in \mathsf{KA}(\mathcal{K}_G) \setminus \mathbf{N}_\omega$, or $\mathsf{OB}_{\mathcal{O},S} \models H$ with $S \subseteq \mathbf{P}_\omega$. By Definition 6.32, we obtain that $\mathbf{K}\, H \in T^u_{\mathcal{K}_G}(\mathbf{P}_\omega, \mathsf{KA}(\mathcal{K}_G) \setminus \mathbf{N}_\omega)$.

Let $\mathbf{K}\, H \in \mathsf{KA}(\mathcal{K}_G) \setminus \mathbf{N}_\omega$, that is $\mathbf{K}\, H \notin \mathbf{N}_\omega$. Then $\mathbf{K}\, H \notin \Gamma'_{\mathcal{K}_G}(\mathbf{P}_\omega)$, i.e., $\mathbf{K}\, H \notin T_{\mathcal{K}_G//\mathbf{P}_\omega} \uparrow \omega$. Thus, two conditions hold. First, for all rules of the form $H \leftarrow \mathcal{B}^+ \wedge \mathcal{B}^-$ in $\mathcal{P}_G$, there is at least one $\mathbf{K}\, A_i \in \mathcal{B}^+$ with $\mathbf{K}\, A_i \in \mathsf{KA}(\mathcal{K}_G) \setminus \mathbf{N}_\omega$ or at least one $\mathbf{not}\, B_j \in \mathcal{B}^-$ with $\mathbf{K}\, B_j \in \mathbf{P}_\omega$, or $\mathsf{OB}_{\mathcal{O},\mathbf{P}_\omega} \models \neg H$. Second, $\mathsf{OB}_{\mathcal{O},\mathbf{N}_\omega} \not\models H$. The first condition corresponds exactly to (Ui) of Definition 6.30 w.r.t. $(\mathbf{P}_\omega, \mathsf{KA}(\mathcal{K}_G) \setminus \mathbf{N}_\omega)$. We derive from the second condition that, for all S with $S \subseteq \mathsf{KA}(\mathcal{K}_G)$ on which $\mathbf{K}\, H$ depends, there is at least one modal $\mathbf{K}$-atom $\mathbf{K}\, A$ such that $\mathsf{OB}_{\mathcal{O},S\setminus\{\mathbf{K}\, A\}} \not\models H$ and $\mathbf{K}\, A$ in $\mathsf{KA}(\mathcal{K}_G) \setminus \mathbf{N}_\omega$. This matches condition (Uii) of Definition 6.30 w.r.t. $(\mathbf{P}_\omega, \mathsf{KA}(\mathcal{K}_G) \setminus \mathbf{N}_\omega)$ and we conclude that $\mathbf{K}\, H \in U_{\mathcal{K}_G}(\mathbf{P}_\omega, \mathsf{KA}(\mathcal{K}_G) \setminus \mathbf{N}_\omega)$.

We have shown that $(\mathbf{P}_\omega, \mathsf{KA}(\mathcal{K}_G) \setminus \mathbf{N}_\omega) \subseteq W_{\mathcal{K}_G}(\mathbf{P}_\omega, \mathsf{KA}(\mathcal{K}_G) \setminus \mathbf{N}_\omega)$. Assume that $(\mathbf{P}_\omega, \mathsf{KA}(\mathcal{K}_G) \setminus \mathbf{N}_\omega)$ is not a fixpoint of $W_{\mathcal{K}_G}$. Then, $W_{\mathcal{K}_G}(\mathbf{P}_\omega, \mathsf{KA}(\mathcal{K}_G) \setminus \mathbf{N}_\omega) = (T_1, F_1)$ such that $\mathbf{P}_\omega \subset T_1$ or $\mathsf{KA}(\mathcal{K}_G) \setminus \mathbf{N}_\omega \subset F_1$. In the first case, $\mathbf{P}_\omega \subset T^u_{\mathcal{K}_G}(\mathbf{P}_\omega, \mathsf{KA}(\mathcal{K}_G) \setminus \mathbf{N}_\omega)$ and there is $\mathbf{K}\, H \in T^u_{\mathcal{K}_G}(\mathbf{P}_\omega, \mathsf{KA}(\mathcal{K}_G) \setminus \mathbf{N}_\omega) \setminus \mathbf{P}_\omega$. Thus, there is a rule of the form $H \leftarrow \mathcal{B}^+ \wedge \mathcal{B}^-$ such that, for all $\mathbf{K}\, A_i \in \mathcal{B}^+$, $\mathbf{K}\, A_i \in \mathbf{P}_\omega$, and, for all $\mathbf{not}\, B_j \in \mathcal{B}^-$, $\mathbf{K}\, B_j \in \mathsf{KA}(\mathcal{K}_G) \setminus \mathbf{N}_\omega$, or $\mathsf{OB}_{\mathcal{O},\mathbf{P}_\omega} \models H$. But then we also have that $\mathbf{K}\, H \in \Gamma_{\mathcal{K}_G}(\mathbf{N}_\omega)$, i.e., $\mathbf{K}\, H \in T_{\mathcal{K}_G/\mathbf{N}_\omega} \uparrow n$ for some n, contradicting the assumption that $\mathbf{P}_\omega$ is a fixpoint. In the second case, $\mathsf{KA}(\mathcal{K}_G) \setminus \mathbf{N}_\omega \subset U_{\mathcal{K}_G}(\mathbf{P}_\omega, \mathsf{KA}(\mathcal{K}_G) \setminus \mathbf{N}_\omega)$ and there is $\mathbf{K}\, H \in U_{\mathcal{K}_G}(\mathbf{P}_\omega, \mathsf{KA}(\mathcal{K}_G) \setminus \mathbf{N}_\omega) \setminus (\mathsf{KA}(\mathcal{K}_G) \setminus \mathbf{N}_\omega)$. Thus, there is a greatest unfounded set w.r.t. $(\mathsf{KA}(\mathcal{K}_G) \setminus \mathbf{N}_\omega)$ that includes $\mathbf{K}\, H$. But then, by Definition 6.30, we also must have $\mathbf{K}\, H \notin \Gamma'_{\mathcal{K}_G}(\mathbf{P}_\omega)$, contradicting the assumption that $\mathbf{N}_\omega$ is a fixpoint.

For the converse, we show that (T, F) is a fixpoint of the operator $W'_{\mathcal{K}_G}$ that is obtained by $W'_{\mathcal{K}_G}(T', F') = (\Gamma_{\mathcal{K}_G}(\mathsf{KA}(\mathcal{K}_G) \setminus F'), \mathsf{KA}(\mathcal{K}_G) \setminus \Gamma'_{\mathcal{K}_G}(T'))$ for any pair of sets of $\mathbf{K}$-atoms (T', F') with $T', F' \in \mathsf{KA}(\mathcal{K}_G)$. This proves that $\mathbf{P}_\omega \subseteq T$ and $\mathsf{KA}(\mathcal{K}_G) \setminus \mathbf{N}_\omega \subseteq F$ since $(\mathbf{P}_\omega, \mathsf{KA}(\mathcal{K}_G) \setminus \mathbf{N}_\omega)$ is the least fixpoint of $W'_{\mathcal{K}_G}$.

We have $(T, F) = W_{\mathcal{K}_G} \uparrow n$ for some n. We show by induction on n that, for $W_{\mathcal{K}_G} \uparrow n = (T_n, F_n)$, $\mathbf{K}\, H \in T_n$ implies $\mathbf{K}\, H \in \Gamma_{\mathcal{K}_G}(\mathsf{KA}(\mathcal{K}_G) \setminus F)$ and $\mathbf{K}\, H \in F_n$ implies $\mathbf{K}\, H \in \mathsf{KA}(\mathcal{K}_G) \setminus \Gamma'_{\mathcal{K}_G}(T)$. This shows that $(T, F) \subseteq W'_{\mathcal{K}_G}(T, F)$.

The base case for $n = 0$ holds trivially. We suppose that the induction is shown for n and we consider two cases for $n + 1$.

Let $\mathbf{K}\, H \in T_{n+1}$. Then $\mathbf{K}\, H \in T^u_{\mathcal{K}_G}(T_n, F_n)$. Thus, there either is a rule of the

form $H \leftarrow \mathcal{B}^+ \wedge \mathcal{B}^-$ in $\mathcal{P}_G$ such that, for all $\mathbf{K}\, A_i \in \mathcal{B}^+$, $\mathbf{K}\, A_i \in T_n$, and, for all $\mathbf{not}\, B_j \in \mathcal{B}^-$, $\mathbf{K}\, B_j \in F_n$, or $\mathsf{OB}_{\mathcal{O},S} \models H$ with $S \subseteq T_n$. By the induction hypothesis and by Definition 6.14, we obtain that $\mathbf{K}\, H \in \Gamma_{\mathcal{K}_G}(\mathsf{KA}(\mathcal{K}_G) \setminus F)$.

Let $\mathbf{K}\, H \in F_{n+1}$. Then $\mathbf{K}\, H \in U_{\mathcal{K}_G}(T_n, F_n)$. Thus, $\mathbf{K}\, U$ occurs in the greatest unfounded set w.r.t. (T_n, F_n). By the induction hypothesis and by Definition 6.18, we obtain that $\mathbf{K}\, H \in \mathsf{KA}(\mathcal{K}_G) \setminus \Gamma'_{\mathcal{K}_G}(T)$.

Finally, we show that (T, F) is in fact a fixpoint of $W'_{\mathcal{K}_G}$. The argument is identical to showing that $(\mathbf{P}_\omega, \mathsf{KA}(\mathcal{K}_G) \setminus \mathbf{N}_\omega)$ is a fixpoint of $W_{\mathcal{K}_G}$ from the other direction in this proof: we assume the contrary and derive for both sets T and F a contradiction to the assumption that (T, F) is a fixpoint of $W_{\mathcal{K}_G}$. □

Note that even though the two (combined) fixpoints are identical, their intermediate iterations are not:

Example 6.36. *Consider $\mathcal{K}_G$.*

$$\mathtt{R} \sqsubseteq \mathtt{S} \tag{6.31}$$

$$\mathbf{K}\, \mathtt{R}(\mathtt{a}) \leftarrow \tag{6.32}$$

$$\mathbf{K}\, \mathtt{p}(\mathtt{a}) \leftarrow \mathbf{not}\, \mathtt{S}(\mathtt{a}) \tag{6.33}$$

We compute the two fixpoints (and we leave **K** *implicit in this presentation).*

$W_{\mathcal{K}_G} \uparrow 0 = (\emptyset, \emptyset)$	$\mathbf{P}_0 = \emptyset$	$\mathbf{N}_0 = \mathsf{KA}(\mathcal{K}_G)$
$W_{\mathcal{K}_G} \uparrow 1 = (\{\mathtt{R}(\mathtt{a})\}, \emptyset)$	$\mathbf{P}_1 = \{\mathtt{R}(\mathtt{a}), \mathtt{S}(\mathtt{a})\}$	$\mathbf{N}_1 = \mathbf{N}_0$
$W_{\mathcal{K}_G} \uparrow 2 = (\{\mathtt{R}(\mathtt{a}), \mathtt{S}(\mathtt{a})\}, \emptyset)$	$\mathbf{P}_2 = \mathbf{P}_1$	$\mathbf{N}_1 = \{\mathtt{R}(\mathtt{a}), \mathtt{S}(\mathtt{a})\}$
$W_{\mathcal{K}_G} \uparrow 3 = (\{\mathtt{R}(\mathtt{a}), \mathtt{S}(\mathtt{a})\}, \{\mathtt{p}(\mathtt{a})\})$	$\mathbf{P}_3 = \mathbf{P}_2$	$\mathbf{N}_3 = \mathbf{N}_2$
$W_{\mathcal{K}_G} \uparrow 4 = W_{\mathcal{K}_G} \uparrow 3$		

Intuitively, the difference is that derivations that do not involve **not***, such as deriving* $\mathbf{K}\, \mathtt{S}(\mathtt{a})$ *from* $\mathbf{K}\, \mathtt{R}(\mathtt{a})$*, are obtained immediately in the same iteration step, i.e., once* $\mathbf{K}\, \mathtt{R}(\mathtt{a})$ *is derived, for the alternating fixpoint, but only in the subsequent iteration for* $W_{\mathcal{K}_G}$*.*

Thus, the two iterations are not identical in general, but the computation of the information that is derived to be false is actually identical if we consider a corresponding pair of sets of **K**-atoms.

Lemma 6.37. *Let $\mathcal{K}_G = (\mathcal{O}, \mathcal{P}_G)$ be a ground hybrid MKNF knowledge base and $(\mathbf{P}_i, \mathbf{N}_i)$ a pair of sets of $\mathbf{K}$-atoms with $T, F \in \mathsf{KA}(\mathcal{K}_G)$ in the computation of the alternating fixpoint of $\mathcal{K}_G$. Then the following holds:*

$$\mathsf{KA}(\mathcal{K}_G) \setminus \mathbf{N}_{i+1} = U_{\mathcal{K}_G}(\mathbf{P}_i, \mathsf{KA}(\mathcal{K}_G) \setminus \mathbf{N}_i)$$

Proof. We show both inclusions from which the equality follows.

$\mathsf{KA}(\mathcal{K}_G) \setminus \mathbf{N}_{i+1} \subseteq U_{\mathcal{K}_G}(\mathbf{P}_i, \mathsf{KA}(\mathcal{K}_G) \setminus \mathbf{N}_i)$: Let $\mathbf{K}\,H \in \mathsf{KA}(\mathcal{K}_G) \setminus \mathbf{N}_{i+1}$. Then $\mathbf{K}\,H \notin \mathbf{N}_{i+1}$, i.e., $\mathbf{K}\,H \notin \Gamma'_{\mathcal{K}_G}(\mathbf{P}_i)$ and $\mathbf{K}\,H \notin T_{\mathcal{K}_G//\mathbf{P}_i} \uparrow \omega$. Thus, two conditions hold. First, for all rules of the form $H \leftarrow \mathcal{B}^+ \wedge \mathcal{B}^-$ in $\mathcal{P}_G$, there is at least one $\mathbf{K}\,A_i \in \mathcal{B}^+$ with $\mathbf{K}\,A_i \in \mathsf{KA}(\mathcal{K}_G) \setminus \mathbf{N}_{i+1}$ or at least one $\mathbf{not}\,B_j \in \mathcal{B}^-$ with $\mathbf{K}\,B_j \in \mathbf{P}_i$, or $\mathsf{OB}_{\mathcal{O},\mathbf{P}_i} \models \neg H$. Second, $\mathsf{OB}_{\mathcal{O},\mathbf{N}_{i+1}} \not\models H$. The first condition corresponds exactly to (Ui) of Definition 6.30 w.r.t. $(\mathbf{P}_i, \mathsf{KA}(\mathcal{K}_G) \setminus \mathbf{N}_i)$. We derive from the second condition that, for all S with $S \subseteq \mathsf{KA}(\mathcal{K}_G)$ on which $\mathbf{K}\,H$ depends, there is at least one modal $\mathbf{K}$-atom $\mathbf{K}\,A$ such that $\mathsf{OB}_{\mathcal{O},S\setminus\{\mathbf{K}\,A\}} \not\models H$ and $\mathbf{K}\,A$ in $\mathsf{KA}(\mathcal{K}_G) \setminus \mathbf{N}_{i+1}$. This matches condition (Uii) of Definition 6.30 w.r.t. $(\mathbf{P}_i, \mathsf{KA}(\mathcal{K}_G) \setminus \mathbf{N}_i)$ and we conclude that $\mathbf{K}\,H \in U_{\mathcal{K}_G}(\mathbf{P}_i, \mathsf{KA}(\mathcal{K}_G) \setminus \mathbf{N}_i)$.

$U_{\mathcal{K}_G}(\mathbf{P}_i, \mathsf{KA}(\mathcal{K}_G) \setminus \mathbf{N}_i) \subseteq \mathsf{KA}(\mathcal{K}_G) \setminus \mathbf{N}_{i+1}$: Let $\mathbf{K}\,H \in U_{\mathcal{K}_G}(\mathbf{P}_i, \mathsf{KA}(\mathcal{K}_G) \setminus \mathbf{N}_i)$. Then $\mathbf{K}\,H$ occurs in the greatest unfounded set w.r.t. $(\mathbf{P}_i, \mathsf{KA}(\mathcal{K}_G) \setminus \mathbf{N}_i)$. It follows from Definition 6.30 that $\mathbf{K}\,H \notin \mathbf{N}_{i+1}$. Consequently, $\mathbf{K}\,H \in \mathsf{KA}(\mathcal{K}_G) \setminus \mathbf{N}_{i+1}$. □

We use the lemma in the next section to simplify some proofs of the properties of the alternating fixpoint and the well-founded MKNF model.

6.4 The Well-Founded MKNF Model and Related Properties

In this section, we prove the properties of the alternating fixpoint construction and the well-founded MKNF model as outlined in Section 6.2. The first such property relates the computed partial partition to truth values in three-valued MKNF models.

The well-founded partition (T_W, F_W) consists of modal atoms that are intended to be true (T_W), false (F_W), or undefined (those modal atoms neither occurring in T_W nor in F_W). But this is not merely an intention. The two sequences of $\mathbf{P}_i$ and $\mathbf{N}_i$ allow us to show that any modal atom that is added to an element of the sequence of $\mathbf{P}_i$, removed from an element of the sequence of $\mathbf{N}_i$ respectively, must be true in all three-valued MKNF models of $\mathcal{K}_G$ (resp. false).

The first proposition of this section indeed establishes that the computed partial partition corresponds to the intended truth values in all three-valued MKNF models of the considered knowledge base $\mathcal{K}$. Note that this result also holds for MKNF-inconsistent knowledge bases.

Proposition 6.38. *Let $\mathcal{K}_G$ be a ground hybrid MKNF knowledge base and (T, F) the pair $(\mathbf{P}_\omega, \mathsf{KA}(\mathcal{K}_G) \setminus \mathbf{N}_\omega)$. Then $\mathbf{K}\,H \in T$ implies that $\mathbf{K}\,H$ is true (and $\mathbf{not}\,H$ is false) in all three-valued MKNF models (M, N) of $\mathcal{K}_G$, and $\mathbf{K}\,H \in F$ implies that $\mathbf{K}\,H$ is false (and $\mathbf{not}\,H$ is true) in all three-valued MKNF models (M, N) of $\mathcal{K}_G$.*

Proof. According to Proposition 6.23, we have to show that, for all i, $\mathbf{K}\,H \in \mathbf{P}_i$ implies that $\mathbf{K}\,H$ is true (and $\mathbf{not}\,H$ is false) in all three-valued MKNF models (M, N) of $\mathcal{K}_G$, and $\mathbf{K}\,H \notin \mathbf{N}_i$ implies that $\mathbf{K}\,H$ is false (and $\mathbf{not}\,H$ is true) in all three-valued MKNF models (M, N) of $\mathcal{K}_G$. We show the argument for $\mathbf{K}\,H$ by an induction on i. This also shows the argument for $\mathbf{not}\,H$ since, for all three-valued MKNF models (M, N) of any given $\mathcal{K}$, we have that $(I, \langle M, N\rangle, \langle M, N\rangle)(\mathbf{K}\,H) = \neg(I, \langle M, N\rangle, \langle M, N\rangle)(\mathbf{not}\,H)$.

The base case $i = 0$ trivially holds, since $\mathbf{P}_0$ is empty and $\mathbf{N}_0$ is equal to $\mathsf{KA}(\mathcal{K}_G)$.

(i) Suppose that the property holds for all $i \leq n$. We consider $i = n + 1$ for two cases, namely $\mathbf{K}\,H \in \mathbf{P}_{n+1}$ and $\mathbf{K}\,H \notin \mathbf{N}_{n+1}$.

Let $\mathbf{K}\,H \in \mathbf{P}_{n+1}$. If $\mathbf{K}\,H$ already occurs in $\mathbf{P}_n$, then $\mathbf{K}\,H$ is true in all three-valued MKNF models (M, N) of $\mathcal{K}_G$, by the induction hypothesis (i). Otherwise, $\mathbf{K}\,H \in \Gamma_{\mathcal{K}_G}(\mathbf{N}_n)$, i.e., $\mathbf{K}\,H \in T_{\mathcal{K}_G/\mathbf{N}_n} \uparrow \omega$ but $\mathbf{K}\,H \notin \mathbf{P}_n$. Since $\mathbf{K}\,H$ is introduced by $T_{\mathcal{K}_G/\mathbf{N}_n} \uparrow \omega$, we know that $\mathbf{K}\,H \in T_{\mathcal{K}_G/\mathbf{N}_n} \uparrow j$ for some j, and we show by induction on j that $\mathbf{K}\,H$ is true in all three-valued MKNF models (M, N) of $\mathcal{K}_G$.

The base case holds trivially, since $T_{\mathcal{K}_G/\mathbf{N}_n} \uparrow 0$ is empty.

(ii) Suppose that the claim holds for all $j \leq m$, and consider $\mathbf{K}\,H \in T_{\mathcal{K}_G/\mathbf{N}_n} \uparrow m+1$.

If $\mathbf{K}\,H$ already occurs in $T_{\mathcal{K}_G/\mathbf{N}_n} \uparrow m$, then the claim holds automatically by the induction hypothesis (ii). Otherwise, there are two cases to consider. Either there is a positive rule $\mathbf{K}\,H \leftarrow \mathbf{K}\,A_1, \ldots \mathbf{K}\,A_n$ in $\mathcal{K}_G/\mathbf{N}_n$ with $\mathbf{K}\,A_i \in T_{\mathcal{K}_G/\mathbf{N}_n} \uparrow m$, or $\mathbf{K}\,H$ is the consequence of $D_{\mathcal{K}_G/\mathbf{N}_n}(T_{\mathcal{K}_G/\mathbf{N}_n} \uparrow m)$. In the first case, by the induction hypothesis (ii), all $\mathbf{K}\,A_i$ are true in all three-valued MKNF models (M, N) of $\mathcal{K}_G$. Additionally, there is a rule $\mathbf{K}\,H \leftarrow \mathbf{K}\,A_1, \ldots \mathbf{K}\,A_n, \mathbf{not}\,B_1, \ldots, \mathbf{not}\,B_m$ in $\mathcal{K}_G$, and since the positive version of this rule occurs in $\mathcal{K}_G/\mathbf{N}_n$, no $\mathbf{K}\,B_j$ occurs in $\mathbf{N}_n$, and thus (by the induction hypothesis (i)), all $\mathbf{K}\,B_j$ are false in all three-valued MKNF models (M, N) of $\mathcal{K}_G$. Consequently, $\mathbf{K}\,H$ has to be true in all three-valued MKNF models (M, N) of $\mathcal{K}_G$. In the second case, $\mathsf{OB}_{\mathcal{O},S} \models H$ with $S = T_{\mathcal{K}_G/\mathbf{N}_n} \uparrow m$ holds. Since $\mathcal{O}$ and all modal atoms occurring in $T_{\mathcal{K}_G/\mathbf{N}_n} \uparrow m$ are true in all three-valued MKNF

models of $\mathcal{K}_G$ (by the induction hypothesis (ii)), we can immediately conclude that $\mathbf{K}\,H$ also has to be true in all three-valued MKNF models (M, N) of $\mathcal{K}_G$.

Alternatively, consider all $\mathbf{K}\,H \notin \mathbf{N}_{n+1}$, i.e., all $\mathbf{K}\,H \notin \Gamma'_{\mathcal{K}_G}(\mathbf{P}_n)$. By Lemma 6.37, $\mathbf{K}\,H \in U_{\mathcal{K}_G}(\mathbf{P}_i, \mathsf{KA}(\mathcal{K}_G) \setminus \mathbf{N}_i)$. By the induction hypothesis (i), we know that, for all $\mathbf{K}\,B \in \mathbf{P}_n$, $\mathbf{K}\,B$ is true for all three-valued MKNF models of $\mathcal{K}_G$. Thus, in case of (Uib), $\mathcal{B}^-$ is false in all three-valued MKNF models of $\mathcal{K}_G$, and, in case of (Uic), $\neg H$ is true in all three-valued MKNF models of $\mathcal{K}_G$. In case of (Uia), the corresponding rule with head $\mathbf{K}\,H$ still appears in the MKNF-coherent transform but some $\mathbf{K}$-atom $\mathbf{K}\,A$ appears in $\mathbf{N}_i$ (and is false in all three-valued MKNF models of $\mathcal{K}_G$ by the induction hypothesis (i)) or part of the greatest unfounded set w.r.t. $(\mathbf{P}_i, \mathsf{KA}(\mathcal{K}_G) \setminus \mathbf{N}_i)$. Likewise, in case of (Uii), some modal $\mathbf{K}$-atom $\mathbf{K}\,A$ occurs in $\mathbf{N}_i$ or part of the greatest unfounded set w.r.t. $(\mathbf{P}_i, \mathsf{KA}(\mathcal{K}_G) \setminus \mathbf{N}_i)$. Each such $\mathbf{K}$-atom is no longer derivable and thus false in all three-valued MKNF models of $\mathcal{K}_G$ since three-valued MKNF models minimize derivable knowledge in the order $\mathbf{t} > \mathbf{u} > \mathbf{f}$.

□

For an MKNF-consistent knowledge base $\mathcal{K}_G$, the pair (T, F) in Proposition 6.38 is defined exactly in the same way as the well-founded partition, and we show below that this correspondence indeed holds. Of course, there is still the issue of determining, based on the iterations and the consistency of $\mathcal{O}$ alone, whether or not the knowledge base is MKNF-consistent. The next theorem presents the necessary and sufficient conditions for MKNF-inconsistency:

Theorem 6.39. *Let $\mathcal{K}_G = (\mathcal{O}, \mathcal{P}_G)$ be a ground hybrid MKNF knowledge base, $\mathbf{P}_\omega$ the fixpoint of the sequence $\mathbf{P}_i$, and $\mathbf{N}_\omega$ the fixpoint of the sequence $\mathbf{N}_i$. $\mathcal{K}_G$ is MKNF-inconsistent iff $\Gamma'_{\mathcal{K}_G}(\mathbf{P}_\omega) \subset \Gamma_{\mathcal{K}_G}(\mathbf{P}_\omega)$ or $\Gamma'_{\mathcal{K}_G}(\mathbf{N}_\omega) \subset \Gamma_{\mathcal{K}_G}(\mathbf{N}_\omega)$ or $\mathcal{O}$ is inconsistent.*

Proof. First, we show that if any of the three conditions holds, then $\mathcal{K}_G$ is MKNF-inconsistent. For the two cases w.r.t. $\mathbf{N}_\omega$ and $\mathbf{P}_\omega$, we present the proof for $\mathbf{N}_\omega$. The other case can be proven analogously.

From Proposition 6.24 we know that $\Gamma_{\mathcal{K}_G}(\mathbf{N}_\omega) = \mathbf{P}_\omega$. Furthermore, by Proposition 6.38, we have that all $\mathbf{K}$-atoms $\mathbf{K}\,H \in \mathbf{P}_\omega$ are true in all three-valued MKNF models (M, N) of $\mathcal{K}_G$. If $\Gamma'_{\mathcal{K}_G}(\mathbf{N}_\omega) \subset \Gamma_{\mathcal{K}_G}(\mathbf{N}_\omega)$, then there is at least one $\mathbf{K}\,H$ such that $\mathbf{K}\,H \in \Gamma_{\mathcal{K}_G}(\mathbf{N}_\omega) \setminus \Gamma'_{\mathcal{K}_G}(\mathbf{N}_\omega)$. The only reason for $\mathbf{K}\,H$ not to occur in $\Gamma'_{\mathcal{K}_G}(\mathbf{N}_\omega)$ is that there is a $\mathbf{K}$-atom $\mathbf{K}\,A$ such that $\mathbf{K}\,A \in \Gamma_{\mathcal{K}_G}(\mathbf{N}_\omega) \setminus \Gamma'_{\mathcal{K}_G}(\mathbf{N}_\omega)$ and $\mathsf{OB}_{\mathcal{O},\mathbf{N}_\omega} \models \neg A$. Either $\mathbf{K}\,H = \mathbf{K}\,A$ or $\mathbf{K}\,H$ and $\mathbf{K}\,A$ appear in a set U that is constructed as in the proof of Proposition 6.38, e.g., for each rule $\mathbf{K}\,H \leftarrow \mathcal{B}$ in $\mathcal{K}_G//\mathbf{N}_\omega$, we have $\mathbf{K}\,A$ in $\mathcal{B}$.

In both cases, $\mathbf{K}\,A$ is true in all three-valued MKNF models of $\mathcal{K}_G$, but the addition of all $\mathbf{K}$-atoms that are not false in all three-valued MKNF models of $\mathcal{K}_G$ (including $\mathbf{K}\,A$) to $\mathcal{O}$ derives $\neg A$. We conclude that $\mathcal{K}_G$ is MKNF-inconsistent.

The third case is a direct consequence of the way evaluation of MKNF formulas is defined: if $\mathcal{O}$ is inconsistent, then there is no first-order model of $\mathcal{O}$. Assume that (M, N) is a three-valued MKNF model of $\mathcal{K}_G$. Then, (M, N) satisfies $\mathcal{K}_G$ and thus also $\mathcal{O}$, i.e., for each $I \in M$, we have $(I, \langle M, N\rangle, \langle M, N\rangle)(\pi(\mathcal{O})) = \mathbf{t}$. Since M must not be empty, we derive a contradiction.

For the other direction, we have to show that any possibly occurring MKNF-inconsistency is detected. So, suppose that $\mathcal{K}_G$ is MKNF-inconsistent. If $\mathcal{O}$ is inconsistent, then we are done immediately. Otherwise, the rules in $\mathcal{P}_G$ alone cannot be MKNF-inconsistent, since they only consist of modal atoms without any appearance of classical negation. Likewise, rules without DL-atoms or rules without DL-atoms in at least some head cannot be inconsistent since the derivation from the ontology $\mathcal{O}$ never conflicts with any rule. Consider thus such an arbitrary DL-atom $\mathbf{K}\,H$ with a rule $\mathbf{K}\,H \leftarrow \mathbf{K}\,A_1, \ldots, \mathbf{K}\,A_n, \mathbf{not}\,B_1, \ldots, \mathbf{not}\,B_m$ in $\mathcal{P}_G$. If H is true as a consequence of $\mathcal{O}$, then the operator $D_{\mathcal{K}_G}$ ensures that $\mathbf{K}\,H$ is true as well, and no inconsistency occurs.

So, let H be first-order false and $\mathbf{K}\,H \in \mathbf{P}_\omega$, i.e., $\mathbf{K}\,H$ is true in all three-valued MKNF models of $\mathcal{K}_G$. But then $\Gamma'_{\mathcal{K}_G}(\mathbf{N}_\omega) \subset \Gamma_{\mathcal{K}_G}(\mathbf{N}_\omega)$ and the inconsistency is detected. Alternatively, $\mathbf{K}\,H$ could be undefined but then $\mathbf{K}\,H \in \mathbf{N}_\omega$, and this is not possible since H is first-order false and $\Gamma'_{\mathcal{K}_G}$ suppresses $\mathbf{K}\,H$. So the only case missing is the one where $\mathbf{K}\,H$ is false in all three-valued MKNF models (as enforced by the operator $\Gamma'_{\mathcal{K}_G}$) but the body of at least one rule with head $\mathbf{K}\,H$ is undefined. Thus $\Gamma'_{\mathcal{K}_G}(\mathbf{P}_\omega) \subset \Gamma_{\mathcal{K}_G}(\mathbf{P}_\omega)$. □

We apply this check for consistency to our previous examples:

Example 6.40. *Consider again $\mathcal{K}_G$ from Example 6.27. We have $\mathbf{P}_\omega = \mathbf{P}_3$ and $\mathbf{N}_\omega = \mathbf{N}_3$. We check for inconsistency (assuming $\mathcal{O}$ is consistent) and obtain $\Gamma'_{\mathcal{K}_G}(\mathbf{P}_\omega) = \Gamma_{\mathcal{K}_G}(\mathbf{P}_\omega)$ and $\Gamma'_{\mathcal{K}_G}(\mathbf{N}_\omega) \subset \Gamma_{\mathcal{K}_G}(\mathbf{N}_\omega)$. So we (rightly) conclude that $\mathcal{K}_G$ is inconsistent.*

Now reconsider $\mathcal{K}_G$ from Example 6.26. We have $\mathbf{P}_\omega = \mathbf{P}_2$ and $\mathbf{N}_\omega = \mathbf{N}_1$. We check for consistency and obtain $\Gamma'_{\mathcal{K}_G}(\mathbf{P}_\omega) = \Gamma_{\mathcal{K}_G}(\mathbf{P}_\omega)$ and $\Gamma'_{\mathcal{K}_G}(\mathbf{N}_\omega) = \Gamma_{\mathcal{K}_G}(\mathbf{N}_\omega)$. Hence, the knowledge base is consistent, and we obtain the well-founded partition

$$(T_W, F_W) = (\{\mathbf{K}\,\mathtt{CD}(\mathtt{Tts}), \mathbf{K}\,\mathtt{int}(\mathtt{Tts}), \mathbf{K}\,\mathtt{Rec}(\mathtt{Tts})\},\\ \{\mathbf{K}\,\mathtt{owns}(\mathtt{Tts}), \mathbf{K}\,\mathtt{LowEv}(\mathtt{Tts})\}).$$

The next example shows that we in fact need both the calculations w.r.t. the two fixpoints.

Example 6.41. *Consider the MKNF-inconsistent knowledge base $\mathcal{K}_G$.*

$$\mathtt{R} \sqsubseteq \neg \mathtt{P} \tag{6.34}$$

$$\mathtt{R}(\mathtt{a}) \tag{6.35}$$

$$\mathbf{K}\,\mathtt{P}(\mathtt{a}) \leftarrow \mathbf{not}\,\mathtt{P}(\mathtt{a}) \tag{6.36}$$

$\mathtt{P}(\mathtt{a})$ *must be false from the ontology alone, and so* $\mathbf{not}\,\mathtt{P}(\mathtt{a})$ *must hold, which immediately causes an inconsistency.*

Only the test using $\mathbf{N}_\omega$ is able to detect this inconsistency. For $\mathsf{KA}(\mathcal{K}_G) = \{\mathbf{K}\,\mathtt{P}(\mathtt{a})\}$, we obtain $\mathbf{P}_\omega = \{\mathbf{K}\,\mathtt{P}(\mathtt{a})\}$ and $\mathbf{N}_\omega = \emptyset$ and thus $\Gamma'_{\mathcal{K}_G}(\mathbf{P}_\omega) = \Gamma_{\mathcal{K}_G}(\mathbf{P}_\omega) = \emptyset$ and $\Gamma'_{\mathcal{K}_G}(\mathbf{N}_\omega) = \emptyset \subset \Gamma_{\mathcal{K}_G}(\mathbf{N}_\omega) = \{\mathbf{K}\,\mathtt{P}(\mathtt{a})\}$.

On the other hand, the following knowledge base is also MKNF-inconsistent, but only the test with $\mathbf{P}_\omega$ allows us to discover this.

$$\mathtt{R} \sqsubseteq \neg \mathtt{P} \tag{6.37}$$

$$\mathtt{R}(\mathtt{a}) \tag{6.38}$$

$$\mathbf{K}\,\mathtt{P}(\mathtt{a}) \leftarrow \mathbf{not}\,\mathtt{u} \tag{6.39}$$

$$\mathbf{K}\,\mathtt{u} \leftarrow \mathbf{not}\,\mathtt{u} \tag{6.40}$$

For $\mathsf{KA}(\mathcal{K}_G) = \{\mathbf{K}\,\mathtt{P}(\mathtt{a}), \mathbf{K}\,\mathtt{u}\}$ we obtain $\mathbf{P}_\omega = \emptyset$ and $\mathbf{N}_\omega = \{\mathbf{K}\,\mathtt{u}\}$ and thus $\Gamma'_{\mathcal{K}_G}(\mathbf{P}_\omega) = \mathbf{N}_\omega \subset \Gamma_{\mathcal{K}_G}(\mathbf{P}_\omega) = \mathsf{KA}(\mathcal{K}_G)$ and $\Gamma'_{\mathcal{K}_G}(\mathbf{N}_\omega) = \Gamma_{\mathcal{K}_G}(\mathbf{N}_\omega) = \mathbf{P}_\omega$.

The difference between the two examples is that in the first example there is a rule with true body and false head, while in the second example there is a rule with undefined body and false head. Each of the two conditions captures one of the cases, which explains why two conditions need to be checked.

As already said, normal rules alone cannot be inconsistent, unless integrity constraints, i.e., rules whose head is $\mathbf{K}\,\mathbf{f}$ (cf. [Motik and Rosati, 2010]), are allowed. In this simpler case, inconsistencies are easily detected since $\mathbf{K}\,\mathbf{f}$ must occur in $\mathsf{KA}(\mathcal{K}_G) \setminus \mathbf{N}_\omega$.

Having established the conditions for checking MKNF-consistency, we can show that the well-founded partition is in fact a partial partition if $\mathcal{K}_G$ is MKNF-consistent.

Proposition 6.42. *Let $\mathcal{K}_G$ be an MKNF-consistent ground hybrid MKNF knowledge base and $(T_W, F_W) = (\mathbf{P}_{\mathcal{K}_G}, \mathsf{KA}(\mathcal{K}_G) \setminus \mathbf{N}_{\mathcal{K}_G})$ the well-founded partition. Then (T_W, F_W) is a partial partition.*

Proof. From Theorem 6.39 and since $\mathcal{K}_G$ is MKNF-consistent, we obtain that $\Gamma'_{\mathcal{K}_G}(\mathbf{P}_\omega) = \Gamma_{\mathcal{K}_G}(\mathbf{P}_\omega)$ and $\Gamma'_{\mathcal{K}_G}(\mathbf{N}_\omega) = \Gamma_{\mathcal{K}_G}(\mathbf{N}_\omega)$. Those two equalities also yield that $T_W \cap F_W = \emptyset$, which shows that (T_W, F_W) is a partition (since T_W and F_W are subsets of $\mathsf{KA}(\mathcal{K}_G)$). □

Not only is the well-founded partition a partition, it can also be shown that the well-founded partition yields an MKNF interpretation pair that satisfies $\mathcal{K}_G$.

Theorem 6.43. *Let $\mathcal{K}_G$ be an MKNF-consistent ground hybrid MKNF knowledge base and $(T_W, F_W) = (\mathbf{P}_{\mathcal{K}_G}, \mathsf{KA}(\mathcal{K}_G) \setminus \mathbf{N}_{\mathcal{K}_G})$ the well-founded partition of $\mathcal{K}_G$. We have that $(I_P, I_N) \models \mathcal{K}_G$ where $I_P = \{I \mid I \models \mathsf{OB}_{\mathcal{O},\mathbf{P}_{\mathcal{K}_G}}\}$ and $I_N = \{I \mid I \models \mathsf{OB}_{\mathcal{O},\mathbf{N}_{\mathcal{K}_G}}\}$.*

Proof. First of all, (I_P, I_N) is a proper MKNF interpretation pair, i.e., since any $I \in I_N$ also satisfies $\mathsf{OB}_{\mathcal{O},\mathbf{P}_{\mathcal{K}_G}}$ we obtain $I_N \subseteq I_P$. By Definition 4.11, we know that $\mathcal{K}_G = \mathbf{K}\,\pi(\mathcal{O}) \wedge \pi(\mathcal{P}_G)$. Since $\pi(\mathcal{O})$ occurs in $\mathsf{OB}_{\mathcal{O},\mathbf{P}_{\mathcal{K}_G}}$ and all $I \in I_P$ satisfy $\mathsf{OB}_{\mathcal{O},\mathbf{P}_{\mathcal{K}_G}}$, we have $(I, \langle I_P, I_N\rangle, \langle I_P, I_N\rangle)(\mathbf{K}\,\pi(\mathcal{O})) = \mathbf{t}$ for all $I \in I_P$. Thus, we only have to consider the evaluation of $(I, \langle I_P, I_N\rangle, \langle I_P, I_N\rangle)(\pi(\mathcal{P}_G))$.

We start by evaluating the modal atoms occurring in $\pi(\mathcal{P}_G)$. Let $\mathbf{K}\,H \in \pi(\mathcal{P}_G)$. Suppose at first that $\mathbf{K}\,H \in T_W$. As such $(I, \langle I_P, I_N\rangle, \langle I_P, I_N\rangle)(\mathbf{K}\,H) = \mathbf{t}$. Alternatively, suppose $\mathbf{K}\,H \in F_W$ and assume that $\mathsf{OB}_{\mathcal{O},\mathbf{N}_{\mathcal{K}_G}} \models H$. In this case, $\mathbf{K}\,H \in \mathbf{N}_{\mathcal{K}_G}$ by means of $\mathcal{D}_{\mathcal{K}_G}$, and we conclude that $\mathsf{OB}_{\mathcal{O},\mathbf{N}_{\mathcal{K}_G}} \not\models H$. Therefore, we have $(I, \langle I_P, I_N\rangle, \langle I_P, I_N\rangle)(\mathbf{K}\,H) = \mathbf{f}$. Finally, let $\mathbf{K}\,H$ occur in $\mathbf{N}_{\mathcal{K}_G}$ but not in T_W or in F_W. We know that $\mathsf{OB}_{\mathcal{O},\mathbf{N}_{\mathcal{K}_G}} \models H$. Assume $\mathsf{OB}_{\mathcal{O},\mathbf{P}_{\mathcal{K}_G}} \models H$. In this case, $\mathbf{K}\,H \in \mathbf{P}_{\mathcal{K}_G}$ by means of $\mathcal{D}_{\mathcal{K}_G}$, and we conclude that $\mathsf{OB}_{\mathcal{O},\mathbf{P}_{\mathcal{K}_G}} \not\models H$. Therefore, we have $(I, \langle I_P, I_N\rangle, \langle I_P, I_N\rangle)(\mathbf{K}\,H) = \mathbf{u}$.

The cases for $\mathbf{not}\,H \in \pi(\mathcal{P})$ proceed analogously. Indeed, if $\mathbf{K}\,H \in T_W$, then we have $(I, \langle I_P, I_N\rangle, \langle I_P, I_N\rangle)(\mathbf{not}\,H) = \mathbf{f}$; if $\mathbf{K}\,H \in T_W$, then $(I, \langle I_P, I_N\rangle, \langle I_P, I_N\rangle)(\mathbf{not}\,H) = \mathbf{t}$ holds; and otherwise $(I, \langle I_P, I_N\rangle, \langle I_P, I_N\rangle)(\mathbf{not}\,H) = \mathbf{u}$ holds.

Now consider $\pi(\mathcal{P}_G)$ which consists of a set of implications, each corresponding to one rule in $\mathcal{P}_G$. To show $(I, \langle I_P, I_N\rangle, \langle I_P, I_N\rangle)(\pi(\mathcal{P}_G)) = \mathbf{t}$, we only have to guarantee that the three cases that map an implication $\supset$ to false do not occur, i.e., the cases where the body of the original rule is true but the head is not (respectively, the body is undefined and the head is false). Assume that any of the three cases holds. If the body of such a rule is true, then by the alternating fixpoint construction we have that the head is true as well, contradicting these two cases. If the rule body is undefined, then (by $\mathbf{N}_{\mathcal{K}_G}$ and the alternating fixpoint) we obtain that the head has to be undefined or true, again contradicting our assumption. Thus, $(I, \langle I_P, I_N\rangle, \langle I_P, I_N\rangle)(\pi(\mathcal{P}_G)) = \mathbf{t}$ holds. □

This result can be combined with Proposition 6.7 to show that the well-founded partition results in a three-valued MKNF model.

Theorem 6.44. *Let $\mathcal{K}_G$ be an MKNF-consistent ground hybrid MKNF knowledge base and $(T_W, F_W) = (\mathbf{P}_{\mathcal{K}_G}, \mathsf{KA}(\mathcal{K}_G) \setminus \mathbf{N}_{\mathcal{K}_G})$ the well-founded partition of $\mathcal{K}_G$. Then (I_P, I_N) is a three-valued MKNF model of $\mathcal{K}_G$, where $I_P = \{I \mid I \models \mathsf{OB}_{\mathcal{O},\mathbf{P}_{\mathcal{K}_G}}\}$ and $I_N = \{I \mid I \models \mathsf{OB}_{\mathcal{O},\mathbf{N}_{\mathcal{K}_G}}\}$.*

Proof. We know from Theorem 6.43 that (I_P, I_N) satisfies $\mathcal{K}_G$. By Proposition 6.7, this MKNF interpretation pair exactly corresponds to the one that equals to a three-valued MKNF model inducing that partition. Thus (I_P, I_N) is a three-valued MKNF model of $\mathcal{K}_G$. □

In fact, (I_P, I_N) is the unique well-founded MKNF model, i.e., the least three-valued MKNF model w.r.t. derivable knowledge.

Theorem 6.45. *Let $\mathcal{K}_G$ be an MKNF-consistent ground hybrid MKNF KB and (I_P, I_N) the three-valued MKNF model of $\mathcal{K}_G$ induced by the well-founded partition (T_W, F_W). For any three-valued MKNF model (M, N) of $\mathcal{K}_G$ we have $(M, N) \succeq_k (I_P, I_N)$. Indeed, (I_P, I_N) is the well-founded MKNF model of $\mathcal{K}_G$.*

Proof. We have shown in Proposition 6.7 that any three-valued MKNF model (M, N) of $\mathcal{K}_G$ induces a partition (T, F) which in turn gives rise to the same three-valued MKNF model (via the objective knowledge). By Proposition 6.38, $\mathbf{K}\,H \in T_W$ implies that $\mathbf{K}\,H$ is true (and $\mathbf{not}\,H$ is false) in all three-valued MKNF models (M, N) of $\mathcal{K}_G$, and $\mathbf{K}\,H \in F_W$ implies that $\mathbf{K}\,H$ is false (and $\mathbf{not}\,H$ is true) in all three-valued MKNF models (M, N) of $\mathcal{K}_G$. We conclude that $T_W \subseteq T$ and $F_W \subseteq F$. Furthermore, we know that $I_P = \{I \mid I \models \mathsf{OB}_{\mathcal{O},T_W}\}$ and $I_N = \{I \mid I \models \mathsf{OB}_{\mathcal{O},\mathsf{KA}(\mathcal{K}_G)\setminus F_W}\}$, and also that $M = \{I \mid I \models \mathsf{OB}_{\mathcal{O},T}\}$ and $N = \{I \mid I \models \mathsf{OB}_{\mathcal{O},\mathsf{KA}(\mathcal{K}_G)\setminus F}\}$. It is straightforward to see that $M \subseteq I_P$ and $I_N \subseteq N$, which by Definition 5.12 finishes the proof. □

This central theorem not only shows that the well-founded MKNF model is unique and well-defined, since the well-founded MKNF model is exactly the three-valued MKNF model that is least w.r.t. $\succeq_k$, but also that the well-founded MKNF model is a sound approximation of any total three-valued MKNF model and therefore of any two-valued MKNF model. Thus, the well-founded partition can also be used in the algorithms presented in [Motik and Rosati, 2010] for computing a subset of the knowledge that holds in all partitions corresponding to a two-valued MKNF model, i.e., for preprocessing the knowledge base in a skeptical way.

Example 6.46. *Consider the simple knowledge base $\mathcal{K}_G$.*

$$\mathtt{P} \sqsubseteq \mathtt{Q} \tag{6.41}$$

$$\mathbf{K}\,\mathtt{P}(\mathtt{a}) \leftarrow \mathbf{K}\,\mathtt{Q}(\mathtt{a}) \tag{6.42}$$

$$\mathbf{K}\,\mathtt{s} \leftarrow \mathbf{not}\,\mathtt{t}, \mathbf{not}\,\mathtt{P}(\mathtt{a}) \tag{6.43}$$

$$\mathbf{K}\,\mathtt{t} \leftarrow \mathbf{not}\,\mathtt{s} \tag{6.44}$$

It is easy to see (and guess and check properly) that there are two two-valued MKNF models: one in which $\mathbf{K}\,\mathtt{s}$ *is true and all other* $\mathbf{K}$*-atoms are false, and one in which* $\mathbf{K}\,\mathtt{t}$ *is true and all other* $\mathbf{K}$*-atoms are false. We may also compute the well-founded partition and obtain that* $\mathbf{K}\,\mathtt{s}$ *and* $\mathbf{K}\,\mathtt{t}$ *are both undefined while the other two* $\mathbf{K}$*-atoms are false. We can thus simplify the KB* $\mathcal{K}_G$ *to the KB* $\mathcal{K}'_G$*.*

$$\mathtt{P} \sqsubseteq \mathtt{Q} \tag{6.45}$$

$$\mathbf{K}\,\mathtt{s} \leftarrow \mathbf{not}\,\mathtt{t} \tag{6.46}$$

$$\mathbf{K}\,\mathtt{t} \leftarrow \mathbf{not}\,\mathtt{s} \tag{6.47}$$

In this concrete example, we reduced the search space for $\mathbf{K}$*-atoms, and even though the example is simple, this kind of preprocessing can be beneficial in general.*

We can also show that the well-founded partition of knowledge bases consisting of only rules, coincides with the well-founded model of the corresponding (normal) logic program as recalled in Chapter 2.

Theorem 6.47. *Let* $\mathcal{K}_G$ *be a ground program of MKNF rules,* Π *a normal logic program obtained from* $\mathcal{P}_G$ *by transforming each MKNF rule*

$$\mathbf{K}\,H \leftarrow \mathbf{K}\,A_1, \ldots, \mathbf{K}\,A_n, \mathbf{not}\,B_1, \ldots, \mathbf{not}\,B_m$$

into a rule

$$H \leftarrow A_1, \ldots, A_n, \mathbf{not}\,B_1, \ldots, \mathbf{not}\,B_m$$

of Π*,* (T_W, F_W) *the well-founded partition of* $\mathcal{K}_G$*, and* W_Π *the well-founded model of* Π*. Then* $\mathbf{K}\,H \in T_W$ *if and only if* $H \in W_\Pi$*, and* $\mathbf{K}\,H \in F_W$ *if and only if* $\mathbf{not}\,H \in W_\Pi$*.*

Finally, the next theorem is obtained from the data complexity results for positive non-disjunctive MKNF knowledge bases in [Motik and Rosati, 2010], where data complexity is measured in terms of A-Box assertions and rule facts.

Theorem 6.48. *Let $\mathcal{K}$ be a hybrid MKNF KB. Assuming that entailment of ground DL-atoms in $\mathcal{DL}$ is decidable with data complexity $\mathcal{C}$, the data complexity of computing the well-founded partition is in $P\text{TIME}^{\mathcal{C}}$.*

For comparison, the data complexity for reasoning with two-valued MKNF models in non-disjunctive programs is shown to be $\mathcal{E}^{\text{PTIME}^{\mathcal{C}}}$ where $\mathcal{E} = \text{NP}$ if $\mathcal{C} \subseteq \text{NP}$, and $\mathcal{E} = \mathcal{C}$ otherwise. Thus, computing the well-founded partition ends up in a strictly smaller complexity class than deriving the two-valued MKNF models. In fact, if the description logic fragment is tractable, then we obtain a formalism whose model is computed with a data complexity in PTIME. This is remarkable, since to the best of our knowledge this is the only tractable local closed world extension for DLs that does not restrict the interaction between the rules and the DL, and that is not limited to stratified rules.

7

Comparison to Related Approaches

As stated in Section 1.5, there are many different approaches for combining rules and ontologies, and many of them are based on completely different ideas. Depending on the applications in mind, the differences may be syntactic, in the sense that they vary on which kinds of rules are used (first-order, non-monotonic, with or without disjunctions in the head, and so on) or on whether the flow of information between the ontology and the rules is bidirectional or not. This defines which predicates are allowed to appear where, and differences in both these syntactic criteria may have an impact on the semantics. But even if two approaches syntactically coincide, the semantics may differ. It may be defined in a modular fashion in which a model consists of two components, one component for the rules and another one for the ontology. Alternatively, a unified semantics may be used interpreting both parts of the combination. Even within each of these two ways of defining a semantics for rules and ontologies, there are differences, which, intuitively, often result from different levels of caution in reasoning. E.g., approaches that are based on the well-founded semantics are usually more skeptical w.r.t. logical consequences than approaches based on stable models.

In this chapter, we compare our semantics to the related work, considering in particular whether the four criteria for a combination of rules and ontologies spelled out in Section 1.4 are satisfied. To make this comparison more easy to read, we add a table to each section summarizing the results. Each table contains, for each approach presented in the section, whether the four criteria are satisfied and, additionally, whether the

semantics of the ontology is used model-wise or in terms of logical consequences.

First, our comparison presents the relation to the two-valued MKNF semantics based on the results obtained in Chapter 6, according to which our semantics can be considered a sound approximation of the two-valued MKNF semantics (Section 7.1). Here, we do not present a table, since both approaches concur on all criteria anyway. The difference is that our semantics is more skeptical in terms of derivable consequences but has a lower computational complexity in general.

Then, several other approaches are presented, which can be divided into approaches that are monotonic (Section 7.2) and those that are non-monotonic and mostly related to the answer set semantics (Section 7.3). For the comparison of these with our work we often rely on the results presented in [Motik and Rosati, 2010] and on the fact that our work is a sound approximation of the two-valued MKNF semantics. In fact, in [Motik and Rosati, 2010] it is shown that hybrid MKNF can capture most of the other major approaches, in the sense that encodings can be provided such that the original knowledge base in the respective approach is satisfiable if and only if the encoding in hybrid MKNF is MKNF satisfiable. Usually, these encodings are based on the more general hybrid MKNF$^+$ knowledge bases that allow us to express first-order knowledge in rules and admit more general modal atoms (see Section 4.4).

Apart from that, there are a few semantics that are not captured by the work in [Motik and Rosati, 2010] but that are of major interest for our work because they are based on the Well-Founded Semantics, namely hybrid rules [Drabent and Małuszyński, 2007, 2010], the well-founded semantics for dl-programs [Eiter et al., 2004, 2011], and the well-founded semantics for normal dl-programs [Lukasiewicz, 2010]. Since our approach is not defined for MKNF$^+$ knowledge bases, we are not able to provide similar technical results showing that our approach captures the former three, but we nevertheless indicate a correspondence based on the results presented in [Motik and Rosati, 2010] (Section 7.4).

There are further approaches that extend DL with non-monotonic reasoning, but that do not rely on rules. For that reason and since most of them are incomparable to our approach in terms of expressiveness, we only mention these here and do not compare them in more detail.

Default Logic [Reiter, 1980] is used in [Baader and Hollunder, 1995] to extend DLs with default rules. It is shown that open defaults (default rules with free variables)

lead to undecidability of the standard reasoning problems, and an restriction of the application of defaults to known individuals is introduced (thus following the same idea that is used for DL-safety) to regain decidability. MKNF logics can also be used to represent the embedding of modal operators into DLs directly. In [Donini et al., 2002], $\mathcal{ALCK}_{\mathcal{NF}}$ is presented that extends the DL $\mathcal{ALC}$ with the operators **K** and **A**, where **A** can be considered semantically equivalent to $\neg$**not**. Defaults, exceptions, and integrity constraints can be expressed directly in the DL and this is used, e.g., in [Grimm and Hitzler, 2008] for semantic matchmaking of Web resources. As spelled out in [Motik and Rosati, 2010], even though hybrid MKNF and $\mathcal{ALCK}_{\mathcal{NF}}$ are both based on MKNF logics, their expressiveness is not compatible. Circumscription [McCarthy, 1980] as formalized in second-order logic can be used as a framework for non-monotonic reasoning [Ferraris et al., 2011]. In [Bonatti et al., 2006, 2009], the computational complexity of reasoning in DLs with circumscription is studied, and several undecidability results are presented together with the restrictions to regain decidability. In [Grimm and Hitzler, 2009], a preferential tableau calculus for $\mathcal{ALCO}$ with circumscription is presented. An extension of [Bonatti et al., 2006, 2009] is provided in [Krisnadhi et al., 2011b], which, in contrast to previous proposals, is applicable to SROIQ without rendering reasoning over the resulting language undecidable.

7.1 Two-valued MKNF semantics

As already stated, the two-valued MKNF semantics satisfies the four criteria for a combination of rules and ontologies, i.e., it is faithful, tight, flexible, and decidable. The same criteria hold for our approach, but there are two differences in the details w.r.t. these criteria.

First, as already pointed out, the two-valued MKNF semantics is defined for a more expressive combination of rules and ontologies, i.e., MKNF$^+$ KBs and general MKNF KBs are considered that only under certain conditions are decidable (see Definition 4.15). Our semantics is only defined for a specific decidable subset of the general MKNF KBs that suits our adaptation of the Well-Founded Semantics. One consequence is that the two-valued MKNF semantics is more expressive and more general. Our semantics can, in principle, be extended to incorporate some of the expressive

features of general hybrid MKNF KBs, such as classical negation in modal atoms occurring in MKNF rules, but then the corresponding constructors have to be available in the considered DL and decidability must not be affected.

Second, by construction, the faithfulness is satisfied for different semantics. Both approaches are faithful to the two-valued first-order semantics of DL languages, but the two-valued MKNF semantics is faithful w.r.t ASP while our approach is faithful w.r.t. the Well-Founded Semantics (Theorem 6.47).

The different underlying LP semantics is also the cause for several further differences between the two MKNF semantics. Just as in the case of SMS and WFS of normal logic programs, there usually exist several two-valued MKNF models that have to be guessed, but there is only one well-founded MKNF model that can be computed. This is the main reason that, in general, our well-founded MKNF semantics has a better data complexity than the two-valued MKNF semantics.

Furthermore, even though there are MKNF-inconsistent KBs, whereas every normal logic program has a well-founded model, there are still KBs for which there exists the well-founded MKNF model but no two-valued MKNF model. This carries over from LP and one simple example is the single MKNF rule $\mathbf{K}\,\mathtt{p} \leftarrow \mathbf{not}\,\mathtt{p}$ with the unique well-founded MKNF model in which p is undefined, but without any two-valued MKNF models.

Moreover, in LP, the WFS is more skeptical than SMS and the same holds for hybrid MKNF knowledge bases. Consider a KB $\mathcal{K}$ consisting of a set of MKNF rules.

$$\mathbf{K}\,\mathtt{p} \leftarrow \mathbf{not}\,\mathtt{q} \tag{7.1}$$

$$\mathbf{K}\,\mathtt{q} \leftarrow \mathbf{not}\,\mathtt{p} \tag{7.2}$$

$$\mathbf{K}\,\mathtt{r} \leftarrow \mathbf{K}\,\mathtt{q} \tag{7.3}$$

$$\mathbf{K}\,\mathtt{r} \leftarrow \mathbf{K}\,\mathtt{p} \tag{7.4}$$

In the well-founded MKNF model of $\mathcal{K}$ all modal atoms appearing in them are undefined, but there are two two-valued MKNF models of $\mathcal{K}$, one in which $\mathbf{K}\,\mathtt{p}$ and $\mathbf{K}\,\mathtt{r}$ are true, and one in which $\mathbf{K}\,\mathtt{q}$ and $\mathbf{K}\,\mathtt{r}$ are true. Thus, $\mathbf{K}\,\mathtt{r}$ is a logical consequence of the two-valued MKNF semantics but not of our well-founded MKNF semantics.

This is intended since the well-founded MKNF model is the least three-valued MKNF model among all three-valued MKNF models, and these include the total three-valued MKNF models corresponding to the two-valued MKNF models. In other words,

Approach	Faithful	Tight	Flexible	Decidable	DL Reasoning
$\mathcal{AL}$-log	yes	no	no	yes	on models
CARIN	yes	no	no	yes	on models
SWRL	yes	yes	no	no	on models
DL-safe Rules	yes	yes	no	yes	on models
DLP	yes	yes	no	yes	on models
DL Rules	yes	yes	no	yes	on models
ELP	yes	yes	no	yes	on models
Nom. Schema	yes	yes	no	yes	on models

Figure 7.1: Comparison of combinations of ontologies with monotonic rules.

the well-founded MKNF semantics is sound w.r.t. the two-valued MKNF semantics (see Theorem 6.45).

7.2 Combinations with First-Order Rules

There exists a large amount of approaches that combine ontologies with first-order rules. Such combinations, by definition, do not allow any non-monotonic reasoning capabilities and are, therefore, not closely related to our work. In fact, none of them satisfies the criterion of flexibility (Section 1.4) and expressing defaults and exceptions is not possible. It is shown in [Motik and Rosati, 2010], that all such first-order combinations of rules and DLs can be captured by special MKNF$^+$ knowledge bases that do not contain any modal atoms in the MKNF$^+$ rules. Such a representation in MKNF$^+$ can then be translated into a hybrid MKNF knowledge base to which the standard reasoning algorithms can be applied [Motik and Rosati, 2010]. We can conjecture that this correspondence carries over to our three-valued semantics since there exists only one two-valued MKNF model for such knowledge bases without modal atoms. For each such model we obtain a corresponding total three-valued MKNF model – the well-founded MKNF model – provided that shifting the respective first-order constructs from rules to the DL is syntactically possible and decidable.

Next, we briefly recall the most important such combinations of ontologies and first-order rules because some of the ideas have been reused for non-monotonic combinations and because some recent work extends the expressiveness of DLs with constructs that

can be translated into rules. Fig. 7.1 summarizes the obtained results. We note that, due to the limitation to monotonic rules, all approaches are faithful, not flexible, and reason on models.

7.2.1 $\mathcal{AL}$-log

One of the first systems combining rules and ontologies is $\mathcal{AL}$-log [Donini et al., 1991, 1998]. In this approach, a knowledge base $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ consists of an ontology in the DL $\mathcal{ALC}$, also called the structural subsystem, and a constrained Datalog program $\mathcal{P}$, also called the relational subsystem. Rules r in $\mathcal{P}$ are function-free, definite rules $\gamma = H \leftarrow A_1, \ldots, A_n$ together with a (possibly empty) set of constraints $C_1(t_1), \ldots, C_m(t_m)$, where each C_j is an $\mathcal{ALC}$ concept description and each t_j is a constant or a variable. Constraints on variables v are intended to restrict the possible values of v, while constraints on constants a impose a condition on a.

Such a knowledge base must also guarantee the following conditions: First, the predicate symbols in the relational and in the structural subsystem must be disjoint; second, the set of used constants must be identical in both, and each constant appearing in $\mathcal{P}$ must appear in $\mathcal{O}$ (if necessary, appropriate assertions may be added); and third, for each constrained rule r of the form $\gamma \& C_1(t_1), \ldots, C_m(t_m)$, if t_j is a variable then t_j appears in γ. By the first condition, it is immediately clear that rules are defined on top of the ontology. In other words $\mathcal{AL}$-log is not a tight approach. The third condition can be interpreted as DL-safety – if there is variable appearing in a construct (normally a DL-atom, here a constraint), then this variable has to appear also in a non-DL atom.

The semantics is defined based on pairs $(\mathcal{I}, \mathcal{H})$ where the first-order interpretation $\mathcal{I}$ interprets $\mathcal{O}$ and the Herbrand interpretation $\mathcal{H}$ handles the (unconstrained) Datalog rules $\mathcal{P}$. Such a pair $(\mathcal{I}, \mathcal{H})$ is a model of $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ if $\mathcal{I}$ is a model for $\mathcal{O}$ and, for each ground $\gamma \& C_1(t_1), \ldots, C_m(t_m)$, either there is $C_j(t_j)$ such that $C_j(t_j)$ is not satisfied by $\mathcal{I}$ or else γ is satisfied by $\mathcal{H}$. Based on this semantics, queries, which are sets containing atoms and constraints, can be answered in $\mathcal{AL}$-log returning a set of variable substitutions. We note that the system does not work simply on logical consequences of the DL but reasons by cases, thus allowing the derivation of information from incomplete knowledge, such as disjunctions. It is shown that query-answering is decidable in $\mathcal{AL}$-log, and a procedure based on resolution is defined [Donini et al., 1998].

7.2.2 CARIN

Another system, CARIN [Levy and Rousset, 1996, 1998], is rather similar to $\mathcal{AL}$-log in the sense that it also combines Datalog rules with an ontology (in the more expressive DL $\mathcal{ALCNR}$) where rules are defined on top of the ontology. Thus, CARIN is not tight either, but this approach is nevertheless closer to a tight integration. Instead of using constraints to include information from the ontology in the rules, ontology predicates are allowed to appear in the body of the Datalog rules, and the semantics is defined using one first-order interpretation that interprets both the ontology and the rules. Such an interpretation is a model of a given combined knowledge base if it satisfies both its components.

For reasoning, the unique name assumption is applied, and the reasoning problem considered is called existential entailment and can be understood as answering (conjunctive) queries. The algorithm itself builds an initial constraint system, and applies propagation rules in a tableau style to obtain one or several non-clashing completions. Then, each such completion can be used to evaluate the Datalog rules for deriving more ground facts. One of the benefits is that the resulting system may allow us to derive information from a rule even though none of its instantiated rule bodies is true.

For general recursive rules, reasoning is undecidable, but without recursion decidability is obtained. Decidability can also be obtained in case of recursive rules, if either certain combinations of DL constructors are not allowed, or by employing role-safe rules. Rules are role-safe if, in every role atom, at least one variable that appears in the role atom also appears in an atom that appears neither in a rule head nor is it constructed using a concept or a role name.

7.2.3 SWRL

Probably the most expressive approach among those that combine ontologies and first-order rules is the Semantic Web Rule Language (SWRL) [Horrocks and Patel-Schneider, 2004; Horrocks et al., 2004]. It combines the ontology language OWL DL with first-order rules in which atoms may be of the form $C(x)$, $P(x, y)$, $\texttt{sameAs}(\texttt{x}, \texttt{y})$, $\texttt{differentFrom}(\texttt{x}, \texttt{y})$, or $\texttt{builtIn}(\texttt{r}, \texttt{x}, ...)$ where C is an OWL description or data range, P is an OWL property, r is a built-in relation, x and y are either variables, OWL individuals or OWL data values, as appropriate. The built-in predicates may be

used to permit arithmetic operations. C may in fact be any complex class description in OWL DL but this can be avoided by introducing new concept names and adding an appropriate equivalence axiom to the ontology. Likewise, `sameAs` and `differentFrom` atoms can be considered syntactic sugar given the expressiveness of the DL. In terms of semantics, the standard first-order semantics of OWL DL is simply extended to rules, which requires adding a specification on variable bindings.

There is no division on the admissible predicates defined in rules, i.e., the approach is a tight combination. Rules have to be safe, which means that variables appearing in the head of the rule have to appear in the body of it, but it is also stated, e.g., in [Horrocks et al., 2004] that, given the expressive power of OWL DL, this is no real limitation of the expressiveness on this tight combination of rules and ontologies. In fact, SWRL is undecidable.

7.2.4 DL-safe Rules

To overcome the problem of undecidability for SWRL, in [Motik et al., 2005], DL-safety is introduced. This is the very same restriction used to maintain decidability for hybrid MKNF knowledge bases. The notion results from a generalization of one of the restrictions on $\mathcal{AL}$-log knowledge bases and, as pointed out in [Motik et al., 2005], it does not restrict the language of each of its two components but merely restricts the interface between them.

DL-safety is obtained by splitting the predicates into DL and non-DL atoms (in the same way as in Definition 4.10) and ensuring that each variable that occurs in a rule also occurs in a non-DL atom in that rule. This guarantees that rules are only applied to individuals appearing explicitly in the knowledge base. The approach remains tight, since DL atoms may appear in the rule head. In practice, DL-safety can be achieved by adding, for each variable x appearing in a rule, a special non-DL atom $O(x)$ to the body of the rule and by adding, for all individuals a that appear in the knowledge base, $O(a)$ to the KB. Alternatively, it can be required that each variable assignment binds all variables only to individuals appearing in the knowledge base.

In [Motik et al., 2005], an algorithm for query answering is defined for a combination of the DL $\mathcal{SHIQ}$ and DL-safe rules, which is based on a reduction of the DL knowledge base to disjunctive logic programs.

7.2.5 DLP

While the reasoning algorithm devised in [Motik et al., 2005] is novel, the idea of translating or reducing a DL fragment into rules is not. In fact, the language DLP [Grosof et al., 2003] (see Section 3.3), which corresponds to the OWL 2 profile OWL RL, is a DL fragment with precise restrictions on the admitted constructors such that it can be translated directly into rules. This obviously reduces the expressiveness but no further restriction to obtain decidability is required. The resulting language is also called def-Horn and it corresponds to function-free Horn logic, which can be considered a generalization of the previously mentioned Datalog programs that not only allows to derive atomic information but also more complex expressions.

Such an approach is, strictly speaking, not a combination of ontologies and rules but a DL fragment that is translatable into rules making it difficult to verify the criteria presented in Section 1.4 as such. However, we may combine a DLP ontology with another DLP ontology already translated into rules and the result is still DLP and a faithful, tight, and decidable combination of ontologies and rules. Thus, the approach satisfies the same criteria as DL-safe rules but the expressiveness is reduced while computational complexity is favorable in comparison.

7.2.6 DL Rules

Given the idea of DLP, one may ask whether it is possible to generalize the idea to more expressive rules. DL Rules [Krötzsch, 2010; Krötzsch et al., 2008a,b] tackles exactly the problem of finding SWRL fragments that are expressible in a certain DL fragment. Each description logic leads to a different set of DL rules, and it is shown that reasoning in the resulting set of rules is in general of the same computational complexity as the corresponding DL fragment. The key idea of DL rules is that the connections of variables in rule bodies are restricted to certain tree-shaped structures. For each binary predicate, the first and the second variable are connected with a directed edge, where each variable corresponds to one vertice in the graph. As an example consider the rule $\mathtt{r}(\mathtt{x},\mathtt{y}) \leftarrow \mathtt{s}(\mathtt{x},\mathtt{y}), \mathtt{p}(\mathtt{y}), \mathtt{t}(\mathtt{x},\mathtt{z})$ in which variables form the tree $\mathtt{y} \leftarrow \mathtt{x} \rightarrow \mathtt{z}$. For a rule that is not tree-shaped, consider $\mathtt{r}(\mathtt{x},\mathtt{y}) \leftarrow \mathtt{s}(\mathtt{x},\mathtt{y}), \mathtt{p}(\mathtt{y},\mathtt{z}), \mathtt{t}(\mathtt{x},\mathtt{z})$ in which there exist two paths from $\mathtt{x}$ to $\mathtt{z}$.

It turns out that allowing role constructors in the DL is very useful to capture fragments of SWRL and, if done carefully, computational complexity is not affected. Important role constructors are concept products ($C \times D$), where C and D are concept expressions, to create roles from concepts directly, or role conjunctions $R \sqcap S$, where R and S are simple roles, that can be used to express rules, such as $\mathtt{T}(\mathtt{x},\mathtt{y}) \leftarrow \mathtt{R}(\mathtt{x},\mathtt{y}), \mathtt{S}(\mathtt{x},\mathtt{y})$, straightforwardly. DL rules, as presented in [Krötzsch, 2010], range from the highly expressive general $\mathcal{SROIQ}(B_s, \times)$ to the tractable $\mathcal{SROEL}(\sqcap_s, \times)$ for which a reasoning algorithm is provided.

Due to the argument presented for DLP, DL rules are also faithful, tight, and decidable.

7.2.7 ELP

Given that DL rules extend the expressiveness of the corresponding fragment, one may consider combining DL rules with DL-safe rules. The result is DL+safe rules [Krötzsch, 2010] and it is shown that, e.g., satisfiability checking in $\mathcal{SROIQ}$+rules is not harder than in $\mathcal{SROIQ}$ itself. So DL+safe rules do not necessarily increase the computational complexity of reasoning but for certain concrete cases, such as tractable DLs, this is no longer the case. In fact, satisfiability checking in DL-safe rules is already NP-complete (see [Krötzsch, 2010]).

A tractable combination of $\mathcal{SROEL}(\sqcap_s, \times)$ with DL-safe rules is obtained by ELP [Krötzsch, 2010; Krötzsch et al., 2008b]. Tractability is achieved by limiting the otherwise arbitrary combinations of atoms in the bodies of DL-safe rules. This can be done by limiting the number of variables occurring in each rule. DL-safety is built directly into the semantics by defining a set of variables that can only be mapped to known individuals, which allows us to regain a tree-shaped structure in the rule body.

ELP is of particular interest since it subsumes both of $\mathcal{SROEL}$ and DLP thus providing a restricted interface between a tractable DL and a tractable DL that can be interpreted as rules, even though the naive combination of these two approaches is intractable. ELP also satisfies all criteria apart from flexibility.

7.2.8 Nominal Schema

ELP is a hybrid approach which means that it is not possible to integrate it directly into existing frameworks defined only for ontology languages. To overcome this problem,

nominal schemas [Krötzsch et al., 2011] have recently been defined. The idea behind nominal schemas is to introduce variables explicitly into DL syntax but restrict their appearance to the syntactic construct of nominals. Then, nominal schemas can be grounded only to individuals occurring in the DL, i.e., such nominal schemas can be considered as nominal variables to which only known individuals can be bound where all occurrences of a nominal schema in one axiom are bound to the same individual. This is quite similar to DL-safe variables only that a nominal schema is a new constructor that seamlessly integrates with DL notation.

For example, a rule such as $\mathtt{C} \leftarrow \mathtt{hasParent}(\mathtt{x},\mathtt{y}), \mathtt{hasParent}(\mathtt{x},\mathtt{z}), \mathtt{isMarried}(\mathtt{y},\mathtt{z})$ that defines the class $\mathtt{C}$ of persons whose parents are married can now be expressed in DLs with nominal schema (and boolean constructors) as follows.

$$\exists \mathtt{hasParent}.\{\mathtt{z}\} \sqcap \exists \mathtt{hasParent}.\exists \mathtt{isMarried}.\{\mathtt{z}\} \sqsubseteq \mathtt{C} \tag{7.5}$$

A general language $\mathcal{SROIQV}(B_s, \times)$ is defined that extends the expressiveness of $\mathcal{SROIQ}$, i.e., OWL 2, and the computational complexity of $\mathcal{SROIQ}$ is maintained. Additionally a tractable language $\mathcal{SROELV}_n(\sqcap, \times)$ is defined where n is the bounded number of unsafe occurrences of nominal schemas in each axiom and a safe occurrence is of a specific syntactic form (see [Krötzsch et al., 2011]). It is also shown that $\mathcal{SROELV}_3(\sqcap, \times)$ suffices to incorporate $\mathcal{SROEL}$ and DLP, i.e., two DLs each of which underlies a tractable profile of OWL 2.

7.3 Ontologies and Non-Monotonic Rules

The approaches in the previous section already require sophisticated restrictions to achieve decidability even though only first-order rules are considered, which by definition are more closely related to DLs than non-monotonic rules. In this section, we recall the most important semantics for combining non-monotonic rules and DLs, which are mostly based on the stable model semantics of LP. Many of these approaches are captured by two-valued hybrid MKNF$^+$ knowledge bases and we point out in each case the correspondence.

Approach	Faithful	Tight	Flexible	Decidable	DL Reasoning
$\mathcal{DL}$+log	yes	yes	no	yes	on models
G-hybrid	yes	yes	no	yes	on models
QEL	yes	yes	no	no	on models
Dl-programs	yes	limited	limited	yes	log. conseq.
Disjunctive dl-programs	limited	yes	yes	yes	on models
Embeddings in AEL	yes	yes	yes	no	log. conseq.
Hybrid MKNF	yes	yes	yes	yes	log. conseq.

Figure 7.2: Comparison of combinations of ontologies with non-monotonic rules based on stable models.

7.3.1 $\mathcal{DL}$+log

One of the first combinations of non-monotonic rules and ontologies is called r-hybrid knowledge bases [Rosati, 2005], which is extended to $\mathcal{DL}$+log [Rosati, 2006]. $\mathcal{DL}$+log combines disjunctive Datalog that also admits the usage of default but not classical negation in rules with an arbitrary (decidable) DL. A knowledge base $\mathcal{K}$ consists of an ontology $\mathcal{O}$ and a set of rules $\mathcal{P}$, where each rule r is of the following form:[1]

$$p_1(\vec{x}_1) \vee \ldots \vee p_n(\vec{x}_n) \leftarrow r_1(\vec{y}_1), \ldots, r_m(\vec{y}_m), s_1(\vec{z}_1), \ldots, s_k(\vec{z}_k), \mathbf{not}\, u_1(\vec{w}_1), \ldots, \mathbf{not}\, u_h(\vec{w}_h)$$

such that $n \geq 0$, $m \geq 0$, $k \geq 0$, and $h \geq 0$, the $\vec{x}_i$, $\vec{y}_i$, $\vec{z}_i$, and $\vec{w}_i$ are tuples of variables and constants, each p_i a DL or a non-DL predicate, each s_i a DL predicate, and each u_i and s_i a non-DL predicate. Additionally, each variable must appear in some $\vec{y}_i$ or $\vec{z}_i$ (rule safety) and every variable appearing in some $\vec{x}_i$ of r must appear in at least one of the $\vec{y}_i$ (weak safety). The latter notion is weaker than DL-safety, since there may exist variables that only appear in a DL-atom in the body of a rule. The standard rule safety ensures nevertheless that there is no variable only appearing in a default negated atom.

The semantics defined in [Rosati, 2006] for $\mathcal{DL}$+log applies the standard name assumption. The non-monotonic semantics[2] considers interpretations, and it is based

[1] We overload the meaning of **not** – here and in the sequel the symbol is used synonymously as default negation and the modal operator in MKNF.

[2] There is also a monotonic semantics defined in [Rosati, 2006], which, due to its monotonic nature is of no interest here.

on the definition of a projection, which intuitively simplifies the program by evaluating the DL-atoms. Given a ground program $\mathcal{P}_G$ and an interpretation $\mathcal{I}$, the projection of $\mathcal{P}_G$ with respect to $\mathcal{I}$, denoted $\Pi(\mathcal{P}_G, \mathcal{I})$, is the ground program obtained from $\mathcal{P}_G$ as follows. For each rule $r \in \mathcal{P}_G$:

- delete r if there exists a DL-atom $p(\vec{t})$ in the head of r such that $\vec{t} \in r^{\mathcal{I}}$;
- delete each DL-atom $p(\vec{t})$ in the head of r such that $\vec{t} \notin r^{\mathcal{I}}$;
- delete r if there exists a DL-atom $s(\vec{t})$ in the body of r such that $\vec{t} \notin s^{\mathcal{I}}$;
- delete each DL-atom $s(\vec{t})$ in the body of r such that $\vec{t} \in s^{\mathcal{I}}$;

The resulting program is free of DL-atoms, and $\mathcal{I}$ is a model of $\mathcal{K}$ if $\mathcal{I}$ is a model of $\mathcal{O}$ and $\mathcal{I}$ restricted to the non-DL predicates is an answer set (see Section 2.4) of $\Pi(\mathcal{P}_G, \mathcal{I})$.

The semantics is faithful, tight, and decidable (due to the safety restrictions) but it is not flexible, even though, unlike the approaches in the previous section, non-monotonic default negation is allowed. However, non-monotonic reasoning is by definition restricted to non-DL atoms, i.e., no default reasoning over information appearing in the DL is possible.

In [Motik and Rosati, 2010], it is shown that hybrid MKNF$^+$ knowledge bases can capture $\mathcal{DL}$+log by transforming each rule into an MKNF$^+$ rule substituting non-DL atoms by modal atoms (atoms A are replaced by $\mathbf{K}\,A$ and atoms with default negation simply maintain the **not**) while DL atoms are transformed into first-order atoms. The MKNF$^+$ KB obtained from such a transformation is satisfiable if and only if the original $\mathcal{DL}$+log KB is satisfiable. It should be noted that this correspondence does not mean that $\mathcal{DL}$+log and hybrid MKNF are equivalent.

Example 7.1. *Consider the knowledge base $\mathcal{K}$, from which, in case of $\mathcal{DL}$+log, we can derive* p.

$$\mathtt{R} \sqsubseteq \mathtt{S} \sqcup \mathtt{T} \tag{7.6}$$

$$\mathtt{R}(\mathtt{a}) \leftarrow \tag{7.7}$$

$$\mathtt{p} \leftarrow \mathtt{S}(\mathtt{a}) \tag{7.8}$$

$$\mathtt{p} \leftarrow \mathtt{T}(\mathtt{a}) \tag{7.9}$$

If we consider the KB as a hybrid MKNF KB and simply substitute all atoms in the rules by modal atoms, then p *is not derivable since neither* S(a) *nor* T(a) *are logical*

consequences of $\mathcal{K}$. Only if we follow the outlined transformation and keep `S(a)` *and* `T(a)` *as first-order atoms, then* `p` *is derivable. Of course, the result of such a transformation is an MKNF$^+$ knowledge base.*

The problem pointed out here is that some approaches, such as our semantics for hybrid MKNF knowledge bases, consider only logical consequences of the DL part, while other semantics, such as $\mathcal{DL}$+log, do consider just any possible model of the DL part.

7.3.2 Guarded Hybrid Knowledge Bases

For $\mathcal{DL}$+log knowledge bases, decidability is achieved by restricting to, respectively, DL-safety and weak DL-safety, and by further restricting non-monotonic reasoning to non-DL atoms. The work on guarded hybrid knowledge bases (g-hybrid knowledge bases) [Heymans et al., 2008a] loosens this restrictions, and is based on open answer set programming [Heymans et al., 2008b].

A g-hybrid knowledge base $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ consists of a DL knowledge base $\mathcal{O}$ and a guarded program $\mathcal{P}$ of rules of the form:

$$H_1 \vee \mathbf{not}\, H_2 \vee \ldots \vee \mathbf{not}\, H_l \leftarrow A_1, \ldots, A_n, \mathbf{not}\, B_1, \ldots, \mathbf{not}\, B_m$$

where all H_k, A_i, and B_j are first-order atoms, and each (positive or negative) part of the rules may be empty. The restriction to one unique positive literal in the head ensures that the adaptation of the GL-transformation results in a non-disjunctive program, while the presence of default negation in the head is required to integrate first-order reasoning into the framework. Decidability is achieved by having one atom A_i (DL or non-DL) function as a guard, i.e., all variables occurring in the rule also occur in the guard. The only exception to that are choice rules of the form $H \vee \mathbf{not}\, H \leftarrow$.

The semantics is based on the open answer set semantics, i.e., Herbrand interpretations with an open domain are considered and the semantics is defined by applying the usual GL-transformation. DL atoms in the rules are preprocessed in the very same way as in the case of $\mathcal{DL}$+log, using the projection. Here, this projection also has to handle the DL atoms with default negation, but it does that by interpreting the default negation classically. A model of the combined KB $\mathcal{K}$ is again a model of $\mathcal{O}$, and an answer set of the result of the (extended) projection. Thus, this approach is also not flexible.

An algorithm is provided that translates the DL language into a guarded program, and in [Heymans et al., 2008a] the non-standard language $\mathcal{DLR}\mathcal{O}^{-\{\leq\}}$, which allows n-ary roles and role conjunctions but no number restrictions, is used as a show case. The approach is less restrictive on the rules but requires the DL to be translatable into guarded programs.

Closely related to g-hybrid KBs is a variant based on forest logic programs [Heymans et al., 2007] that is called f-hybrid knowledge bases [Feier and Heymans, 2009]. In forest logic programs, rules are syntactically restricted to a form that is tree-shaped and as a result the guard can be dropped from the rules without affecting decidability. F-hybrid knowledge bases combine a $\mathcal{SHOQ}$ knowledge base and a forest logic program. The semantics is based on the same first-order projection and open answer sets, and for reasoning the $\mathcal{SHOQ}$ is translated into a forest logic program.

7.3.3 Quantified Equilibrium Logics

The approaches presented so far in this section are all variants of the same main idea for combining non-monotonic rules and ontologies, only varying on the restrictions applied to ensure decidability and the domain used. In [de Bruijn et al., 2007b], quantified equilibrium logics (QEL) [Pearce and Valverde, 2005] is adapted to incorporate r-hybrid and g-hybrid knowledge bases and $\mathcal{DL}$+log into one unifying framework.

Knowledge bases $\mathcal{K}$ consist of an ontology $\mathcal{O}$ and a program of rules of the form

$$H_1 \vee \ldots \vee H_k \vee \neg H_{k+1} \vee \ldots \vee \neg H_l \leftarrow A_1, \ldots, A_n, \neg B_1, \ldots, \neg B_m$$

where all H_i, A_i, and B_i are function-free, first-order atoms. The negation in the head is interpreted classically if the negated atom is a DL atom, and non-monotonically otherwise. To obtain the semantics, the stable closure of $\mathcal{K}$ is defined by adding, for all DL predicates P, $\forall\vec{x}.(P(\vec{x}) \vee \neg P(\vec{x}))$ to $\mathcal{K}$, and each of the semantics is obtained by varying the notion of a model in QEL.

As such, the approach is not flexible either, but, in general, it is also undecidable since decidability is not a primary concern in [de Bruijn et al., 2007b]. Instead, the unifying framework is provided, and for decidability the solutions applied in each of the captured approaches are referenced.

7.3.4 dl-programs

An entirely different approach is called dl-programs [Eiter et al., 2008b]. In this approach, a knowledge base $\mathcal{K}$ consists again of a DL knowledge base $\mathcal{O}$ and a program $\mathcal{P}$ containing (non-disjunctive) dl-rules of the form

$$H \leftarrow A_1, \leftarrow A_n, \mathbf{not}\, B_1, \ldots, \mathbf{not}\, B_m$$

where H is a first-order atom or a classically negated atom, and all A_i and B_j are first-order atoms, classically negated atoms, or special dl-atoms to query the ontology. Note that the sets of predicates is divided into disjoint sets of DL predicates, which only appear in dl-atoms and in $\mathcal{O}$, and non-DL predicates, so that dl-atoms are the only means of transferring information between the ontology and the rules. Such dl-atoms are of the form

$$DL[S_1\ op_1\ p_1, \ldots, S_l\ op_l\ p_l; Q](\vec{t})$$

where S_i are DL predicates, p_i are non-DL predicates, $op_i \in \{\uplus, \cup, \cap\}$, Q is an n-ary DL predicate, and $\vec{t}$ a vector of n terms. The intuitive idea is that we query the ontology for $Q(\vec{t})$ where certain ontology predicates S_i are altered by information derived in the rules. Intuitively, $\uplus$ is used to augment S_i with the derived knowledge from p_i; $\cup$ augments $\neg S_i$ with the derived knowledge from p_i and $\cap$ augments $\neg S$ with what is not derived in p_i.

The semantics of dl-programs is based on the answer set semantics using Herbrand interpretations. To apply the answer set semantics to dl-programs the only open problem is the interpretation of the dl-atoms. For that, a dl-query $Q(t)$ is defined to be either (a) a GCI F or its negation $\neg F$; (b) of the forms $C(t)$ or $\neg C(t)$, where C is a concept, and t is a term; (c) of the forms $R(t_1, t_2)$ or $\neg R(t_1, t_2)$, where R is a role, and t_1 and t_2 are terms; or (d) of the forms $= (t_1, t_2)$ and $\neq (t_1, t_2)$, where t_1 and t_2 are terms. Note that, in the case of (a), t is empty. Given an Herbrand interpretation I, a ground dl-atom $DL[\lambda; Q](\vec{c})$ with $\lambda = S_1\ op_1\ p_1, \ldots, S_l\ op_l\ p_l$ is satisfied in I iff $\mathcal{O}(I; \lambda) = \mathcal{O} \cup \bigcup_{i=1}^{m} A_i(I) \models Q(\vec{c})$, and, for $1 \leq i \leq m$,

$$A_i(I) = \begin{cases} \{S_i(\vec{e}) \mid p_i(\vec{e}) \in I\}, & \text{if } op_i = \uplus; \\ \{\neg S_i(\vec{e}) \mid p_i(\vec{e}) \in I\}, & \text{if } op_i = \cup; \\ \{\neg S_i(\vec{e}) \mid p_i(\vec{e}) \notin I\}, & \text{if } op_i = \cap. \end{cases}$$

As a simple example consider that the rules contain a fact $\mathtt{p}(\mathtt{a})$ and the DL an axiom $\mathtt{C} \sqsubseteq \mathtt{D}$. Then, a dl-atom $DL[\mathtt{C} \uplus \mathtt{p}; \mathtt{D}](\mathtt{a})$ can be derived to be true. Thus, dl-atoms are used as interfaces between the ontology and rules: information derivable in the DL can be queried for in the rules, while information in the rules can be passed to the DL via λ. However, this transfer is limited in the sense that it is not stored. This means that if a certain piece of information from the rules is to be used for each dl-query, then it has to be added explicitly each time in the corresponding dl-atom.

DL-atoms can be split into monotonic ones – those containing only $\uplus$ and $\dot{\cup}$ – and non-monotonic ones – those containing at least one occurrence of $\dot{\cap}$ since an increasing set p_i would decrease derivable consequences on $\neg S_i$. Two different answer set semantics are defined depending on whether all dl-atoms or only the non-monotonic ones are treated by the GL-transformation. In the strong answer set semantics the transformation is only extended to reduce non-monotonic dl-atoms, while in the weak answer set semantics all dl-atoms are removed from the transform.

It is also shown that weak answer sets are not minimal due to possibly occurring loops in the monotonic dl-atoms, and it is argued in [Eiter et al., 2008b] that weak answer sets are less preferable but may still be a reasonable choice if no information about the monotonicity of dl-atoms is available. Moreover, in [Wang et al., 2010] it is shown that this problem carries over to strong answer sets of dl-programs. Consider a knowledge base $\mathcal{K} = (\emptyset, \mathcal{P})$ where $\mathcal{P}$ contains the rule:

$$\mathtt{p}(\mathtt{a}) \leftarrow DL[\mathtt{c} \uplus \mathtt{p}, \mathtt{b} \dot{\cap} \mathtt{q}; \mathtt{c} \sqcap \neg \mathtt{b}](\mathtt{a}). \tag{7.10}$$

Then, both $\emptyset$ and $\{\mathtt{p}(\mathtt{a})\}$ are strong (and weak) answer sets where the latter is not intended. This is the result of the fact that the circular dependency between $\mathtt{c}$ and $\mathtt{p}$ is removed in the transform. The solution to that problem is either using canonical answer sets based on loop formulas [Wang et al., 2010] or not to use the operator $\dot{\cap}$ at all.

In [Eiter et al., 2008b], dl-programs are defined for the DLs underlying OWL DL and OWL Lite. But the approach is applicable to any decidable DL. The approach itself is faithful and decidable, but only to some extent tight and flexible. As argued in [Motik and Rosati, 2010], rules cannot derive new facts about DL predicates. They can only pose conditional queries to the DL. Moreover, we can never derive non-monotonic consequences for DL predicates directly. Instead, we have to use the interfaces, which

may appear under default negation. Additionally, there are indirect effects of using the non-monotonic operator $\not\supset$, but only if we do not disallow it to avoid the problems just described. Thus, the approach is only flexible with some limitations. Nevertheless, the work is of considerable importance, also due to the fact that it has been generalized in hex-programs [Eiter et al., 2006] where interfaces can be created to arbitrary external sources.

Finally, in [Motik and Rosati, 2010], a transformation from dl-programs $\mathcal{K}$ to MKNF knowledge bases $\mathcal{K}'$ allowing for arbitrary modal atoms is defined, and it is shown that $\mathcal{K}$ has a strong answer set if and only if $\mathcal{K}'$ has a two-valued MKNF model. The transformation is restricted to knowledge bases without $\not\supset$, and it transforms all rule atoms into modal atoms using **K** and **not**, where dl-atoms A are transformed as follows:

$$\tau(A) = \left(\pi(\mathcal{O}) \wedge \bigwedge \tau(S_i op_i p_i)\right) \supset Q(\vec{t}) \qquad \text{where}$$

$$\tau(S_i, op_i, p_i) = \begin{cases} \forall \vec{x} : p_i(\vec{x}) \supset S_i(\vec{x}) & \text{if } op_i = \uplus \\ \forall \vec{x} : p_i(\vec{x}) \supset \neg S_i(\vec{x}) & \text{if } op_i = \uplus \end{cases}$$

Since $\mathcal{O}$ is included in the translation of each dl-atom, we obtain that $\mathcal{K}' = (\emptyset, \mathcal{P}')$.

7.3.5 Disjunctive dl-Programs

Though similar in name, disjunctive dl-programs [Lukasiewicz, 2007, 2010] and their rationale are completely different from dl-programs. In fact, the work aims at using vocabulary from formal ontologies in rule-based systems without any restrictions, and the idea to achieve this is to view the integration from the perspective of rules rather than from DLs as it is usually done.

In this approach, a knowledge base $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ consists of an ontology $\mathcal{O}$ and a disjunctive program without classical negation as before but, in particular, without any DL-atoms. There is no separation of the predicates, i.e., one alphabet is used for the entire KB.

The semantics is straightforwardly defined, following the idea of viewing the integration from the perspective of the rules. Only Herbrand interpretations are considered.[1] Such an Herbrand interpretation I is a model of a DL knowledge base $\mathcal{O}$ if and only if

[1] To be precise, the approach considers finite domains that may contain more elements than those appearing in $\mathcal{K}$ but without loss of generality, we can assume that the domain is exactly the set of all constants appearing in $\mathcal{K}$.

$\mathcal{O} \cup I \cup \{\neg a \mid a \in HB_{\mathcal{P}_G} \setminus I\}$ is satisfiable. This means that the ontology is not used to derive arbitrary knowledge, but that it is precisely defined to verify only atoms and their default negations w.r.t. the ontology. Given a disjunctive dl-program $\mathcal{K} = (\mathcal{O}, \mathcal{P})$, I is a model of $\mathcal{K}$ if and only if I is a model of $\mathcal{O}$ and $\mathcal{P}$, and I is an answer set if I is a minimal model of the usual GL-like transform.

The approach does, indeed, not consider any restrictions such as DL-safety or forest-structures on the considered KBs. But for decidability in rules, it requires that the set of considered constants is finite, i.e., in the algorithm a finite subset I of $HB_{\mathcal{P}_G}$ is considered, and it is verified whether I is a minimal model. Consequently, the approach is tight, flexible, and decidable, but only faithful to some extent [Motik and Rosati, 2010]. The entailment relation is only defined for atoms, but not, e.g., for classically negated atoms. This is not really surprising given that I is not used for derivations apart from atoms.

A technically advanced encoding of disjunctive dl-programs in MKNF is given (see Definition 7.3 in [Motik and Rosati, 2010]) showing that a disjunctive dl-program $\mathcal{K}$ has an answer set if and only if its MKNF encoding is MKNF satisfiable.

7.3.6 Embeddings in Autoepistemic Logic

Another approach related to hybrid MKNF is based on first-order autoepistemic logic (AEL) [Konolige, 1991; Moore, 1985]. Several embeddings of non-ground logic programs into first-order AEL are defined and the results are lifted from logic programs to combinations of first-order theories, such as DLs, and logic programs [de Bruijn et al., 2007a, 2011].

No computational properties are known, and, in [Motik and Rosati, 2010], it is conjectured that the encoding of propositional AEL into MBNF logics [Rosati, 1999] can be easily adapted to MKNF and that it might be possible to establish a similar encoding for first-order AEL and use it to provide an algorithm in MKNF for these embeddings. However, no concrete result has been established yet. It seems that the embedding $\tau_{EH}^{\vee}$ with epistemic rule bodies and epistemic rule heads and, additionally without unique name assumption, is the one most closely related to hybrid MKNF, not only syntactically but also with respect to the semantic consequences, but even that is not a precise correspondence (see [de Bruijn et al., 2011]).

Approach	Faithful	Tight	Flexible	Decidable	DL Reasoning
Hybrid Rules	yes	no	no	yes	on models
Dl-programs	yes	limited	limited	yes	log. conseq.
Normal dl-programs	limited	yes	yes	yes	log. conseq.
Hybrid MKNF	yes	yes	yes	yes	log. conseq.

Figure 7.3: Comparison of combinations of ontologies with non-monotonic rules based on well-founded semantics.

7.4 Combinations based on the Well-Founded Semantics

There are only a few approaches combining rules and ontologies based on the Well-Founded Semantics and we recall them in this section. Just like our semantics, each of the three approaches is based on some work we already presented in this chapter, and, in all but one case, this related approach is based on stable models. We also verify whether the four criteria on combinations of rules and ontologies hold to compare the three semantics in particular to our semantics.

7.4.1 Hybrid Rules with Well-founded Semantics

The approach of hybrid rules with well-founded semantics [Drabent and Małuszyński, 2007, 2010] is motivated by $\mathcal{AL}$-log of [Donini et al., 1998] (see Section 7.2) and can be considered as an extension of that work. The approach of hybrid rules generalizes $\mathcal{AL}$-log to normal rules with constraints and arbitrary function symbols and considers more general theories than just DLs, but the decidable fragment focuses again on DLs and function-free atoms in rules. Next, we recall the technical details from [Drabent and Małuszyński, 2010] concentrating on the decidable fragment.

Predicates are divided into two disjoint sets of non-DL predicates and DL predicates, while the same set of constants is used for both the DL and the rules. A knowledge base $\mathcal{K}$ consists of an ontology $\mathcal{O}$ in some DL and a program $\mathcal{P}$ containing rules of the form

$$H \leftarrow C, A_1, \ldots, A_n, \mathbf{not}\, B_1, \ldots, \mathbf{not}\, B_m$$

where H, A_i, and B_j are atoms (over non-DL predicates) and C is a constraint (over DL-predicates). Constraints are conjunctive queries to the DL.

For the semantics, Herbrand interpretations are considered. More precisely, it considers signed sets, i.e., interpretations do not only contain the atoms that are true, but also explicitly those that are false. Given $(\mathcal{O}, \mathcal{P})$, let M be a model of $\mathcal{O}$.[1] Let $\mathcal{P}/M$ be the normal program obtained from $\mathcal{P}_G$ by

- removing each rule constraint C which is true in M;
- removing each rule whose rule constraint is not true in M.

The well-founded model of $\mathcal{P}/M$ is called the well-founded model of P based on M. Intuitively, the constraints are evaluated w.r.t. some first-order model M and for the resulting normal program the well-founded semantics of LP is applied. Thus, there is not one unique well-founded model but, depending on the chosen M, several models for each $\mathcal{K}$ giving this semantics a distinct flavor that is somewhat closer to a stable model semantics approach. One result of that is that the logical consequence relation allows for four truth values, **t**, **u**, **f**, and a forth one for conflicting truth values as the result of different first-order models M of the constraints. E.g., a DL assertion $(\mathtt{C} \sqcup \mathtt{D})(\mathtt{a})$ and a constrained rule $\mathtt{p}(\mathtt{a}) \leftarrow \mathtt{C}(\mathtt{a})$ yield that $\mathtt{p}(\mathtt{a})$ is true or false depending on whether $\mathtt{C}(\mathtt{a})$ is modeled in the considered first-order model M. In such cases, the provided top-down procedure returns a set of cases each with its associated constraints. This top-down procedure is based on a generalization of SLS-resolution [Przymusinski, 1989] and uses XSB in its realization [Drabent et al., 2007]. It is shown that the procedure is sound and complete, and decidability is achieved by a notion similar to DL-safety taking into account that constraints may contain equality. No complexity results are provided.

Constraints are used as interfaces to the ontology, and since these do only appear in the body of rules, it is immediately obvious, that hybrid rules with well-founded semantics is not a tight approach. Moreover, since default negation is not allowed in constraints, the approach is not flexible either. Nevertheless, the combination is faithful to the first-order semantics of DLs and the WFS of logic programs.

Finally, the approach is capable of deriving $\mathtt{p}$ in Example 7.1 where $\mathtt{S}(\mathtt{a})$ and $\mathtt{T}(\mathtt{a})$ are constraints to the ontology. There is a model M_1 of $\mathcal{O}$ such that $\mathtt{S}(\mathtt{a})$ is true in M_1 and a model M_2 of $\mathcal{O}$ such that $\mathtt{T}(\mathtt{a})$ is true in M_2 and in both normal programs $\mathcal{P}/M_1$ and $\mathcal{P}/M_2$, p is true.

[1] In [Drabent and Małuszyński, 2010], it is referred to the usual first-order semantics for models of a DL.

7.4.2 Well-founded semantics for dl-programs

In [Eiter et al., 2004, 2011] a well-founded semantics for dl-programs is provided. The work considers precisely the setting of dl-programs [Eiter et al., 2008b] as presented in Section 7.3.4, but without classical negation and the non-monotonic operator $\not\in$, which, as we have seen, causes some problems in the case of the answer set semantics. The justification is that $\not\in$ can be rewritten using $\uplus$ and the complement of the considered predicate (see [Eiter et al., 2011]).

The semantics is defined using an extension of the unfounded sets construction (cf. Definition 2.25). The notion defined for the approach in [Eiter et al., 2011] keeps the conditions known from LP and it adds two conditions dealing with dl-atoms. One condition says that a dl-atom A_i is never true if we expand I in such a way that all unfounded atoms are kept false. The other condition says, for a dl-atom B_j, that $\mathbf{not}\, B_j$ is definitely false, regardless of how I is expanded. Now, if we compare this unfounded set notion to the unfounded set style presentation of three-valued MKNF (shown in the proof of Proposition 6.38), then it is apparent, due to the tighter integration of DLs and rules in our work, that the conditions for three-valued MKNF are considerably more complicated. For dl-programs, the well-founded model is defined based on iterations and unfounded sets, very much in the same way as in Definition 2.27 for LP.

It is shown that the semantics is faithful w.r.t. WFS of LP and w.r.t. the first-order semantics, but limited for the latter to the consequence relation used in the interfaces to the DL. A corresponding alternating fixpoint is defined, and it is shown that the semantics approximates the strong answer set semantics for dl-programs (without $\not\in$). Combined complexity and data complexity results are presented for $\mathcal{SHIF}(\mathbf{D})$ and $\mathcal{SHOIN}(\mathbf{D})$, the DL languages corresponding to OWL DL and OWL Lite. A more general data complexity result corresponds exactly to our Theorem 6.48. This means that for a tractable DL fragment, dl-programs also maintain a polynomial data complexity. An implementation is devised based on the alternating fixpoint using DLV and RACER and details can, e.g., be found in [Eiter et al., 2008b].

The classification w.r.t. the four criteria on combinations of rules and ontologies carry over from Section 7.3.4, i.e., the approach is faithful and decidable but only to a limited extend tight and flexible. However, it is pointed out in [Eiter et al., 2011] that these limitations may be beneficial when extending the approach to more general

variants of entailment, and when considering inconsistency which is limited here to inconsistency of $\mathcal{O}$.

With respect to Example 7.1, we obtain a corresponding dl-program where the atoms `S(a)` and `T(a)` are turned into dl-atoms. This semantics does not derive `p` since the interfaces in dl-programs operate on logical consequences of the DL. In other words, only logical consequences of the DL considered.

7.4.3 Well-founded semantics for normal dl-programs

The only other well-founded semantics [Lukasiewicz, 2010] is defined for normal dl-programs, which form a subclass of disjunctive dl-programs [Lukasiewicz, 2007, 2010] (see Section 7.3.5). The syntax is exactly the same as for disjunctive dl-programs with the only difference that disjunctions in the heads of the rules are not allowed.

The semantics uses a similar notion of unfounded sets as dl-programs, even though the two approaches are considerably different. In fact, the unfounded set condition of [Eiter et al., 2011] regarding (positive) dl-atoms is quite similar to the one used for normal dl-programs. The main difference is that, in [Lukasiewicz, 2010], only the positive part $\mathcal{O}^+$ of the ontology is considered for reasoning. For that, it is required that the ontology is decomposable into a positive and a negative part (cf. the notions w.r.t. Datalog$^{\pm}$ [Calì et al., 2009]). This severely restricts the approach, but in [Lukasiewicz, 2010] the *DL-Lite* family is presented as an example of DL languages that are decomposable in that way. The well-founded model is again obtained from operators similar to Definition 2.27 where the operator T also derives information from $\mathcal{O}^+$.

It is shown in [Lukasiewicz, 2010] that the defined semantics is faithful to the WFS of LP, and an algorithm based on a corresponding alternating fixpoint operator is defined. It is also shown that the semantics is a sound approximation of the answer set semantics of normal dl-programs. Complexity results for $DL\text{-}Lite_{\mathcal{A}}$[1] [Poggi et al., 2008] are provided stating that the combined complexity is EXP and the data complexity is P. So, this is another semantics that for a (at least one) tractable DL language yields a polynomial data complexity for the combination of the DL with normal rules.

[1] $DL\text{-}Lite_{\mathcal{A}}$ combines $DL\text{-}Lite_{\mathcal{R}}$ with functionality statements from $DL\text{-}Lite_{\mathcal{F}}$ but limits some interaction of these since otherwise the computational complexity of $DL\text{-}Lite_{\mathcal{R}}$ is lost (see Section 3.3.2).

Just like disjunctive dl-rules, the approach is tight, flexible, and decidable but only faithful to the WFS of LP and to the DL w.r.t. positive atoms (as presented in Section 7.3.5).

Interestingly, while the answer set semantics for disjunctive dl-programs allows us to derive p in Example 7.1, the same does not hold for the well-founded semantics for normal dl-programs. This means that the interaction with the DL changes from a model-based view to one of logical consequences (but limited to positive atoms), while dl-programs and hybrid MKNF (not considering MKNF$^+$ KBs) both follow the view of logical consequences in both approaches (WFS and answer sets).

7.4.4 Comparison

Summing up the collected information, we have identified approaches that combine nonmonotonic rules and ontologies based on the well-founded semantics as the preferable ones. Combinations of ontologies and monotonic rules do not allow to express defaults and exceptions, and approaches based on stable models are slightly more expressive in terms of derivable information, but have, in general, a worse computational complexity.

Among all combinations of ontologies and rules based on the well-founded semantics, our work is the only one that satisfies the four criteria we presented in Section 1.4 and the most expressive combination of ontologies and rules. Normal dl-programs only allow reasoning on atomic information in the DL, while dl-programs limit the interaction between the rules and the ontology to special interfaces that do not allow to transfer information from the rules to the ontology permanently. The approach of hybrid rules is even less tight, since rules are simply defined on top of a DL.

The limitations in expressiveness, and the fact that the semantics of hybrid rules relies on several well-founded models, makes this approach the weakest one from our point of view. Not only is its idea to have several well-founded models counterintuitive, it also makes it more difficult to actually compute answers in terms of complexity, since it is not enough to consider one computation but all w.r.t. the possible models of the DL.

The approach for normal dl-programs is probably the most straightforward integration if we consider DL such as *DL-Lite* that are decomposable. However, due to this particular restriction, the approach is not applicable to a large number of DL, making

it only a good choice if the desired knowledge representation and reasoning capabilities fit with decomposable languages, such as $DL\text{-}Lite_{\mathcal{A}}$.

Just like our semantics, the well-founded semantics for dl-programs is applicable to any arbitrary decidable DL and the interaction with the DL is based on logical consequences from the DL interacting with the rules. The difference lies in the definition of the interaction between the rules and the ontology. In the case of dl-programs, it is limited to specific interfaces. This has some technical benefits since actual implementations are easier to realize and to generalize to a wider set of external sources of information. However, the interaction is cumbersome on a syntactical level. No predicate can appear in the rules and in the ontology simultaneously. To ensure a correspondence, additional rules have to be added and in each dl-atom the information of the rules has to be added to the ontology. This is particularly impractical for modeling large knowledge bases: each dl-atom may contain a long list of transfers. In our approach, we simply allow atoms to appear everywhere, so that we do not require special interfaces to transfer information from the rules to the ontology and vice versa. Thus, we achieve an approach that is more expressive, flexible, and tight, yet easier to use for modeling information.

Part III

Querying Hybrid MKNF Knowledge Bases

8

SLG($\mathcal{O}$)– A General Querying Procedure

The alternating fixpoint in Chapter 6 provides a way to compute, in a naive bottom-up fashion, a representation of all the consequences of a knowledge base. However, such an approach is impractical for large knowledge bases especially if we are only interested in certain parts of the information. Consider, e.g., the medical case study [Patel et al., 2007] (as presented in Chapter 1): knowledge of whether a specific patient is using a certain medication does not require knowledge of the medications of thousands of other patients. In general, despite its polynomial complexity, bottom-up computation of the well-founded MKNF model does not scale to enterprise applications, much less to those of the Semantic Web. A query-driven procedure corresponding to our new semantics, which only consults information relevant for a specific patient, would clearly be preferable in this case.

In this chapter, we present such a querying mechanism, called **SLG($\mathcal{O}$)**, that is sound and complete for the well-founded MKNF model, and sound for MKNF knowledge bases w.r.t. the semantics of Motik and Rosati [2010]. **SLG($\mathcal{O}$)** accepts DL-safe conjunctive queries, i.e., conjunctions of atoms with variables, where queries have to be ground when processed in the ontology, returning all correct answer substitutions for variables in the query.

To ease the definition of such a querying mechanism, we start by presenting an alternative way for the bottom-up computation that doubles the knowledge base by creating a slightly modified variant of the original knowledge base, so that only one

operator is used for the alternating fixpoint construction (Section 8.1). This complicates the presentation on the syntactic level but the top-down computation becomes easier in the sense that we do not have to decide when to use which of the two alternating operators. We could alternatively use the computation based on unfounded sets so that we avoid this decision. But that computation does not admit the detection of inconsistencies which is why we rely on this transformation that doubles the knowledge base.

Then, we extend SLG resolution with tabling [Chen and Warren, 1996] with the possiblity to query an accompanying ontology (Section 8.2). We show that the new procedure is sound and complete w.r.t. the well-founded MKNF model and that the procedure terminates and, under certain conditions, maintains the favorable computational complexity of that model (Section 8.3).

8.1 Alternative Bottom-Up Iteration

As presented in Chapter 6, the bottom-up computation of the well-founded MKNF model, more precisely the well-founded partition, requires two operators. The difference between these two operators is on whether or not the derivability of the classical negation of the rule head is used to eliminate a rule from the respective transform. Using both operators directly would complicate the top-down procedure, since we would have to use them alternately in different parts of the procedure. Limiting the computation to only one of the two operators is not an option either, given that $\Gamma_{\mathcal{K}}$ sometimes yields counterintuitive results, while $\Gamma_{\mathcal{K}}$ alone would hide possible inconsistencies (see Section 6.2).

Thus, in this section we define that computation in a different way. Instead of computing the two separate sequences $\mathbf{P}_i$ and $\mathbf{N}_i$ based on different operators we provide a transformation of $\mathcal{K}$ that allows us to compute the two sequences using only one operator. Intuitively, we provide a knowledge base that allows us to separate the contexts of true and non-false derivations on a syntactic level. Then, we define one operator that can be used to compute both $\mathbf{P}_i$ and $\mathbf{N}_i$ while still being able to ensure coherence and detecting possible inconsistencies. Concretely, we transform the original knowledge base $\mathcal{K}$, by doubling the rules and the ontology in $\mathcal{K}$ using new predicates. The resulting knowledge base can be used for computing both sequences. We only have

to keep two separate renamings of the ontology to ensure that a possible inconsistency in one sequence (see, e.g., the computation of $\mathbf{N}_1$ in Example 6.27) does not affect the consistency of the other. This does not cause any problems in the forthcoming top-down procedure, since the transformation will be defined such that we know which renaming of the ontology we have to query.

The following definition introduces two new special predicates for each predicate appearing in $\mathcal{K}$ based on which the transformation that doubles a knowledge base $\mathcal{K}$ is defined.

Definition 8.1. *Let $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ be a hybrid MKNF knowledge base. We introduce new predicates A^d and NA for each predicate A appearing in $\mathcal{K}$, and define $\mathcal{O}^d$ from $\mathcal{O}$ by substituting each predicate A in $\mathcal{O}$ by A^d, and $\mathcal{P}^d$ by transforming each rule $\mathbf{K}\, H(t_i) \leftarrow \mathbf{K}\, A_1, \ldots, \mathbf{K}\, A_n, \mathbf{not}\, B_1, \ldots, \mathbf{not}\, B_m$ occurring in $\mathcal{P}$, t_i representing the arguments of H, into two rules:*

(1) $\mathbf{K}\, H(t_i) \leftarrow \mathbf{K}\, A_1, \ldots, \mathbf{K}\, A_n, \mathbf{not}\, B_1^d, \ldots, \mathbf{not}\, B_m^d$ *and either*

(2a) $\mathbf{K}\, H^d(t_i) \leftarrow \mathbf{K}\, A_1^d, \ldots, \mathbf{K}\, A_n^d, \mathbf{not}\, B_1, \ldots, \mathbf{not}\, B_m, \mathbf{not}\, NH(t_i)$ *if H is a DL-atom; or*

(2b) $\mathbf{K}\, H^d(t_i) \leftarrow \mathbf{K}\, A_1^d, \ldots, \mathbf{K}\, A_n^d, \mathbf{not}\, B_1, \ldots, \mathbf{not}\, B_m$ *otherwise*

We define the doubled hybrid MKNF knowledge base $\mathcal{K}^d = (\mathcal{O}, \mathcal{O}^d, \mathcal{P}^d)$.

Intuitively, the atoms based on the original predicates A represent truth while the atoms based on the newly introduced predicates A^d represent non-falsity, i.e., we use, e.g., $A^d(a)$ to represent the non-falsity of $A(a)$. The new atoms $NH(t_i)$ appearing in (2a), are used as a marker to distinguish between rules that may be affected by the derivability of the classical negation of its head (as in $\Gamma'_{\mathcal{K}}$) and the others (as in $\Gamma_{\mathcal{K}}$). This marker has to be referenced in the modified transform, but before that, we define a slightly different operator for doubled positive hybrid MKNF knowledge bases that takes into account the parallel computations on the two renamings of the ontology.

Definition 8.2. *Let $\mathcal{K}_G^d = (\mathcal{O}, \mathcal{O}^d, \mathcal{P}_G^d)$ be a doubled positive, ground hybrid MKNF knowledge base. The operators $R_{\mathcal{K}_G^d}$, $D_{\mathcal{K}_G^d}$, and $T_{\mathcal{K}_G^d}$ are defined on subsets of $\mathsf{KA}(\mathcal{K}_G^d)$*

as follows:

$$R_{\mathcal{K}_G^d}(S) = \{\mathbf{K}\,H \mid \mathcal{P}_G^d \textit{ contains a rule of the form (6.7)} \\ \textit{such that, for all } i, 1 \leq i \leq n, \mathbf{K}\,A_i \in S\}$$

$$D_{\mathcal{K}_G^d}(S) = \{\mathbf{K}\,\xi \mid \mathbf{K}\,\xi \in \mathsf{KA}(\mathcal{K}_G) \textit{ and } \mathsf{OB}_{\mathcal{O},S} \models \xi\} \cup \\ \{\mathbf{K}\,\xi \mid \mathbf{K}\,\xi \in \mathsf{KA}(\mathcal{K}_G^d) \setminus \mathsf{KA}(\mathcal{K}_G) \textit{ and } \mathsf{OB}_{\mathcal{O}^d,S} \models \xi\}$$

$$T_{\mathcal{K}_G^d}(S) = R_{\mathcal{K}_G^d}(S) \cup D_{\mathcal{K}_G^d}(S)$$

This definition is the same as Definition 6.9 apart from two differences. First, the doubled knowledge base $\mathcal{K}_G^d$ is considered and, consequently, **K**-atoms from $\mathsf{KA}(\mathcal{K}_G^d)$ appear. Second, the operator $D_{\mathcal{K}_G^d}$ calculates consequences from $\mathcal{O}$ and $\mathcal{O}^d$ in parallel but limited to the corresponding set of atoms appearing in each of the two renamings, thus preventing any unintended interaction due to possible inconsistencies. This could be further refined by stating that only an appropriate part of the set S is added to $\mathcal{O}$ and $\mathcal{O}^d$, namely the part that can appear in $\mathcal{O}$ and $\mathcal{O}^d$ respectively. In terms of derivable consequences, adding, e.g., S containing A^d to $\mathcal{O}$ is the same as adding $S \setminus \{A^d\}$ to $\mathcal{O}$ since atoms based on doubled predicates never have any impact on $\mathcal{O}$, and the same holds for the atoms based on original predicates and $\mathcal{O}^d$. We avoid formalizing this to keep the notation simple. As already pointed out before, in practice only one ontology $\mathcal{O}$ can be used for the computation of $D_{\mathcal{K}_G^d}(S)$ if we calculate the two parts one after the other and keep the appropriate contexts, i.e., we may take all atoms from S that are based on doubled predicates, add only these to $\mathcal{O}$ using the original predicates, and return all derivable **K**-atoms using the doubled predicates.

Next, we present a slightly altered version of the MKNF-coherent transform (cf. Definition 6.17) taking the special doubled hybrid MKNF knowledge base and the new **not**-atoms that serve as markers into account.

Definition 8.3. *Let $\mathcal{K}_G^d = (\mathcal{O}, \mathcal{O}^d, \mathcal{P}_G^d)$ be a doubled, ground hybrid MKNF knowledge base and $S \subseteq \mathsf{KA}(\mathcal{K}_G^d)$. The* MKNF^d-coherent transform $\mathcal{K}_G^d//'S$ *is defined as $\mathcal{K}_G^d//'S = (\mathcal{O}, \mathcal{O}^d, \mathcal{P}_G^d//'S)$, where $\mathcal{P}_G^d//'S$ contains all rules*

$$\mathbf{K}\,H \leftarrow \mathbf{K}\,A_1, \ldots, \mathbf{K}\,A_n$$

for which there exists a rule

$$\mathbf{K}\, H \leftarrow \mathbf{K}\, A_1, \ldots, \mathbf{K}\, A_n, \mathbf{not}\, B_1, \ldots, \mathbf{not}\, B_m$$

in $\mathcal{P}_G^d$ *with* $\mathbf{K}\, B_j \notin S$ *for all* $1 \leq j \leq m$ *and* $\mathsf{OB}_{\mathcal{O},S} \not\models \neg H'$ *if* $\mathbf{not}\, NH'$ *appears in the body.*

This definition is almost identical to Definition 6.17 with the only difference that the removal of a rule with head $\mathbf{K}\, H$ due to the derivation of $\neg H'$ is only possible in marked rules. Given Definition 8.1, this means that only rules with head $\mathbf{K}\, H^d$ may be eleminated if $\neg H$ can be derived from $\mathcal{O}$ and S. In a top-down system the marker itself can be used to actually trigger a query to the ontology for $\neg H'$ or to introduce appropriate correspondences to the ontology that link $\neg H'$ with NH'.

We can now define a new operator $\Gamma^d_{\mathcal{K}_G}$ for ground knowledge bases $\mathcal{K}_G$ similar to $\Gamma_{\mathcal{K}_G}$ in Definition 6.14 and $\Gamma'_{\mathcal{K}_G}$ in Definition 6.18, which operates on $\mathbf{K}$-atoms of $\mathsf{KA}(\mathcal{K}_G^d)$.

Definition 8.4. *Let* $\mathcal{K}_G^d = (\mathcal{O}, \mathcal{O}^d, \mathcal{P}_G^d)$ *be a doubled, ground hybrid MKNF knowledge base and* $S \subseteq \mathsf{KA}(\mathcal{K}_G^d)$*. We define:*

$$\Gamma_{\mathcal{K}_G^d}(S) = T_{\mathcal{K}_G^d//'S} \uparrow \omega.$$

We can show that this operator is antitonic just as its two predecessors.

Lemma 8.5. *Let* $\mathcal{K}_G^d$ *be a doubled, ground hybrid MKNF knowledge base and* $S_1 \subseteq S_2 \subseteq \mathsf{KA}(\mathcal{K}_G^d)$*. Then* $\Gamma_{\mathcal{K}_G^d}(S_2) \subseteq \Gamma_{\mathcal{K}_G^d}(S_1)$*.*

Proof. We have to show that $T_{\mathcal{K}_G^d//'S_2} \uparrow \omega \subseteq T_{\mathcal{K}_G^d//'S_1} \uparrow \omega$. Since $\mathcal{K}_G$ is finite, $\mathcal{K}_G^d$ is also finite, and we prove by induction on n that, for all $n < \omega$, $T_{\mathcal{K}_G^d//'S_2} \uparrow n \subseteq T_{\mathcal{K}_G^d//'S_1} \uparrow n$ holds.

The base case for $n = 0$ is trivial since $\emptyset \subseteq \emptyset$.

Assume that $T_{\mathcal{K}_G^d//'S_2} \uparrow n \subseteq T_{\mathcal{K}_G^d//'S_1} \uparrow n$ holds, consider $\mathbf{K}\, H \in T_{\mathcal{K}_G^d//'S_2} \uparrow (n+1)$. Then $\mathbf{K}\, H \in T_{\mathcal{K}_G^d//'S_2}(T_{\mathcal{K}_G^d//'S_2} \uparrow n)$ and there are two cases to consider:

1. $\mathcal{K}_G^d//'S_2$ contains a rule of the form $\mathbf{K}\, H \leftarrow \mathbf{K}\, A_1, \ldots, \mathbf{K}\, A_n$ such that $\mathbf{K}\, A_i \in T_{\mathcal{K}_G^d//'S_2} \uparrow n$ for each $1 \leq i \leq n$. In this case, since $S_1 \subseteq S_2$ holds, we also have $\mathbf{K}\, H \leftarrow \mathbf{K}\, A_1, \ldots, \mathbf{K}\, A_n$ in $\mathcal{K}_G^d//'S_1$ and, by the induction hypothesis, $\mathbf{K}\, A_i \in T_{\mathcal{K}_G^d//'S_1} \uparrow n$ holds for each $1 \leq i \leq n$. Hence, $\mathbf{K}\, H \in T_{\mathcal{K}_G^d//'S_1} \uparrow (n+1)$.

2. $\mathbf{K}\,H$ is a consequence obtained from $D_{\mathcal{K}_G^d}$. But $D_{\mathcal{K}_G^d}$ derives only consequences from the unchanged DL renamings $\mathcal{O}$ and $\mathcal{O}^d$ together with $T_{\mathcal{K}_G^d//'S_2} \uparrow n$. By the induction hypothesis, we conclude that $\mathbf{K}\,H \in T_{\mathcal{K}_G^d//'S_1} \uparrow (n+1)$. □

Since this new operator is antitonic, we can define its iteration similar to Definition 6.21, only now we have just one operator.

Definition 8.6. *Let $\mathcal{K}_G^d$ be a doubled, ground hybrid MKNF knowledge base. We define:*

$$\mathbf{P}_0^d = \emptyset \qquad \mathbf{N}_0^d = \mathsf{KA}(\mathcal{K}_G^d)$$
$$\mathbf{P}_{n+1}^d = \Gamma_{\mathcal{K}_G^d}(\mathbf{N}_n^d) \qquad \mathbf{N}_{n+1}^d = \Gamma_{\mathcal{K}_G^d}(\mathbf{P}_n^d)$$
$$\mathbf{P}_\omega^d = \bigcup \mathbf{P}_n^d \qquad \mathbf{N}_\omega^d = \bigcap \mathbf{N}_n^d$$

The correspondence between Definitions 6.21 and 8.6 can be established with a precise relation between the atoms in the doubled knowledge base $\mathcal{K}_G^d$ and those in $\mathcal{K}_G$. To ease the proof of the corresponding property, we adapt the notion of unfounded sets (Definition 6.30) to provide a result similar to Lemma 6.37, which relates unfounded sets and the sequence $\mathbf{N}_i^d$.

For that purpose, we also adapt the notion of dependency.

Definition 8.7. *Let $\mathcal{K}_G = (\mathcal{O}, \mathcal{O}^d, \mathcal{P}_G^d)$ be a doubled, ground hybrid MKNF knowledge base, H an atom with $H \in \mathsf{KA}(\mathcal{K}_G^d)$, and S a (possibly empty) set of atoms with $S \subseteq \mathsf{KA}(\mathcal{K}_G^d)$. We say that H* depends *on S if and only if, for $\mathcal{O}' = \mathcal{O}$ or $\mathcal{O}' = \mathcal{O}^d$:*

(i) $\mathsf{OB}_{\mathcal{O}',S} \models H$ *and*

(ii) there is no S' with $S' \subset S$ such that $\mathsf{OB}_{\mathcal{O}',S'} \models H$.

Based on this notion of dependency, the notion of an unfounded set for hybrid MKNF is extended from Definition 6.30 to include $\mathcal{O}$ and $\mathcal{O}^d$ of the doubled KB.

Definition 8.8. *Let $\mathcal{K}_G^d = (\mathcal{O}, \mathcal{O}^d, \mathcal{P}_G^d)$ be a doubled ground hybrid MKNF knowledge base and (T, F) a pair of sets of $\mathbf{K}$-atoms with $T, F \subseteq \mathsf{KA}(\mathcal{K}_G^d)$. We say that $U \subseteq \mathsf{KA}(\mathcal{K}_G^d)$ is an* unfounded set *(of $\mathcal{K}_G^d$)* with respect to (T, F) *if, for each $\mathbf{K}$-atom $\mathbf{K}\,H \in U$, the following conditions are satisfied:*

(Ui) for each rule $\mathbf{K}\,H \leftarrow \mathcal{B}$ in $\mathcal{P}_G$ at least one of the following holds.

(Uia) Some $\mathbf{K}$-atom $\mathbf{K}\,A$ appears in $\mathcal{B}$ and in $U \cup F$.

(Uib) Some **not***-atom* **not** *B appears in $\mathcal{B}$ and in T.*

(Uic) $\mathsf{OB}_{\mathcal{O},T} \models \neg H_1(t_i)$ *and* $\mathbf{not}\, H_1(t_i) \in \mathcal{B}$*, where* $H = H_1^d(t_i)$

(Uii) for each (possibly empty) S *on which* $\mathbf{K}\, H$ *depends, with* $S \subseteq \mathsf{KA}(\mathcal{K}_G)$ *and* $\mathsf{OB}_{\mathcal{O},S}$ *consistent, there is at least one modal* $\mathbf{K}$*-atom* $\mathbf{K}\, A$ *such that* $\mathsf{OB}_{\mathcal{O},S\setminus\{\mathbf{K}\, A\}} \not\models H$ *and* $\mathbf{K}\, A$ *in* $U \cup F$*.*

(Uii') for each (possibly empty) S *on which* $\mathbf{K}\, H$ *depends, with* $S \subseteq \mathsf{KA}(\mathcal{K}_G^d)$ *and* $\mathsf{OB}_{\mathcal{O}^d,S}$ *consistent, there is at least one modal* $\mathbf{K}$*-atom* $\mathbf{K}\, A$ *such that* $\mathsf{OB}_{\mathcal{O}^d,S\setminus\{\mathbf{K}\, A\}} \not\models H$ *and* $\mathbf{K}\, A$ *in* $U \cup F$*.*

The union of all unfounded sets of $\mathcal{K}_G^d$ *w.r.t.* (T, F) *is called the* greatest unfounded set *of* $\mathcal{K}_G^d$ *w.r.t.* (T, F) *and denoted* $U_{\mathcal{K}_G^d}(T, F)$*.*

The only differences in comparison to Definition 6.30 is that (Uic) is adapted to $\mathcal{K}_G^d$ and that a variant of (Uii) is introduced to handle derivations w.r.t. $\mathcal{O}^d$.

The correspondence to the sequence $\mathbf{N}_i^d$ is established as in Lemma 6.37.

Lemma 8.9. *Let* $\mathcal{K}_G^d = (\mathcal{O}, \mathcal{O}^d, \mathcal{P}_G^d)$ *be a doubled ground hybrid MKNF knowledge base and* $(\mathbf{P}_i^d, \mathbf{N}_i^d)$ *a pair of sets of* $\mathbf{K}$*-atoms with* $(\mathbf{P}_i^d, \mathbf{N}_i^d \in \mathsf{KA}(\mathcal{K}_G^d)$ *in the computation of the alternating fixpoint of* $\mathcal{K}_G^d$*. Then the following holds:*

$$\mathsf{KA}(\mathcal{K}_G^d) \setminus \mathbf{N}_{i+1}^d = U_{\mathcal{K}_G^d}(\mathbf{P}_i^d, \mathsf{KA}(\mathcal{K}_G^d) \setminus \mathbf{N}_i^d)$$

Proof. We show both inclusions from which the equality follows.

$\mathsf{KA}(\mathcal{K}_G^d) \setminus \mathbf{N}_{i+1}^d \subseteq U_{\mathcal{K}_G^d}(\mathbf{P}_i^d, \mathsf{KA}(\mathcal{K}_G^d) \setminus \mathbf{N}_i^d)$: Let $\mathbf{K}\, H \in \mathsf{KA}(\mathcal{K}_G^d) \setminus \mathbf{N}_{i+1}^d$. Then $\mathbf{K}\, H \notin \mathbf{N}_{i+1}^d$, i.e., $\mathbf{K}\, H \notin \Gamma_{\mathcal{K}_G^d}(\mathbf{P}_i^d)$ and $\mathbf{K}\, H \notin T_{\mathcal{K}_G^d//'\mathbf{P}_i^d} \uparrow \omega$. Thus, two conditions hold. First, for all rules of the form $H \leftarrow \mathcal{B}^+ \wedge \mathcal{B}^-$ in $\mathcal{P}_G^d$, there is at least one $\mathbf{K}\, A_i \in \mathcal{B}^+$ with $\mathbf{K}\, A_i \in \mathsf{KA}(\mathcal{K}_G^d) \setminus \mathbf{N}_{i+1}^d$ or at least one $\mathbf{not}\, B_j \in \mathcal{B}^-$ with $\mathbf{K}\, B_j \in \mathbf{P}_i^d$, or $\mathsf{OB}_{\mathcal{O},\mathbf{P}_i} \models \neg H_1(t_i)$ and $\mathbf{not}\, H_1(t_i) \in \mathcal{B}^-$, where $H = H_1(t_i)$. Second, neither $\mathsf{OB}_{\mathcal{O},\mathbf{N}_{i+1}^d} \models H$ nor $\mathsf{OB}_{\mathcal{O}^d,\mathbf{N}_{i+1}^d} \models H$ holds. The first condition corresponds exactly to (Ui) of Definition 8.8 w.r.t. $(\mathbf{P}_i^d, \mathsf{KA}(\mathcal{K}_G^d) \setminus \mathbf{N}_i^d)$. We derive from the second condition that, for all S with $S \subseteq \mathsf{KA}(\mathcal{K}_G^d)$ on which $\mathbf{K}\, H$ depends, there is at least one modal $\mathbf{K}$-atom $\mathbf{K}\, A$ such that $\mathsf{OB}_{\mathcal{O},S\setminus\{\mathbf{K}\, A\}} \not\models H$ as well as $\mathsf{OB}_{\mathcal{O}^d,S\setminus\{\mathbf{K}\, A\}} \not\models H$ and $\mathbf{K}\, A$ in $\mathsf{KA}(\mathcal{K}_G^d) \setminus \mathbf{N}_{i+1}^d$. This matches condition (Uii) of Definition 8.8 w.r.t. $(\mathbf{P}_i^d, \mathsf{KA}(\mathcal{K}_G^d) \setminus \mathbf{N}_i^d)$ and we conclude that $\mathbf{K}\, H \in U_{\mathcal{K}_G^d}(\mathbf{P}_i^d, \mathsf{KA}(\mathcal{K}_G^d) \setminus \mathbf{N}_i^d)$.

$U_{\mathcal{K}_G^d}(\mathbf{P}_i^d, \mathsf{KA}(\mathcal{K}_G^d) \setminus \mathbf{N}_i^d) \subseteq \mathsf{KA}(\mathcal{K}_G^d) \setminus \mathbf{N}_{i+1}^d$: Let $\mathbf{K}\, H \in U_{\mathcal{K}_G^d}(\mathbf{P}_i^d, \mathsf{KA}(\mathcal{K}_G^d) \setminus \mathbf{N}_i^d)$. Then $\mathbf{K}\, H$ occurs in the greatest unfounded set w.r.t. $(\mathbf{P}_i^d, \mathsf{KA}(\mathcal{K}_G^d) \setminus \mathbf{N}_i^d)$. It follows from Definition 8.8 that $\mathbf{K}\, H \notin \mathbf{N}_{i+1}^d$. Consequently, $\mathbf{K}\, H \in \mathsf{KA}(\mathcal{K}_G^d) \setminus \mathbf{N}_{i+1}^d$. □

We can now show the correspondence between modal atoms of $\mathcal{K}_G$ and $\mathcal{K}_G^d$.

Proposition 8.10. *Let $\mathcal{K}_G = (\mathcal{O}, \mathcal{P}_G)$ be a ground hybrid MKNF knowledge base. Then the following holds:*

- $\mathbf{K}\, A \in \mathbf{P}_\omega$ *if and only if* $\mathbf{K}\, A \in \mathbf{P}^d_\omega$.
- $\mathbf{K}\, B \notin \mathbf{N}_\omega$ *if and only if* $\mathbf{K}\, B^d \notin \mathbf{N}^d_\omega$.

Proof. We show by induction on n that two conditions hold.

(i) $\mathbf{K}\, A \in \mathbf{P}_n$ if and only if $\mathbf{K}\, A \in \mathbf{P}^d_n$

(ii) $\mathbf{K}\, B \notin \mathbf{N}_n$ if and only if $\mathbf{K}\, B^d \notin \mathbf{N}^d_n$

This is sufficient since the grounded knowledge base is finite, which means that the iteration is finite and stops for some natural number n, i.e., the two fixpoints coincide on the relevant atoms as in (i) and (ii).

The base case for $n = 0$ is straightforward since $\mathbf{P}_0$ and $\mathbf{P}^d_0$ are empty while $\mathbf{N}_0$ and $\mathbf{N}^d_0$ both contain their entire Herbrand base.

(1) So, suppose that (i) and (ii) hold for n and let $\mathbf{K}\, A \in \mathbf{P}_{n+1}$ and $\mathbf{K}\, B \notin \mathbf{N}_{n+1}$. We show that $\mathbf{K}\, A \in \mathbf{P}^d_{n+1}$ and $\mathbf{K}\, B^d \notin \mathbf{N}^d_{n+1}$. The other direction of the equivalence follows from an identical argument.

(i) First, suppose that $\mathbf{K}\, A \in \mathbf{P}_{n+1}$ but $\mathbf{K}\, A \notin \mathbf{P}_n$ (otherwise we obtain the result by the induction hypothesis (1) immediately). We show that $\mathbf{K}\, A \in \mathbf{P}^d_{n+1}$. If $\mathbf{K}\, A \in \mathbf{P}_{n+1}$, then $\mathbf{K}\, A \in \Gamma_{\mathcal{K}}(\mathbf{N}_n)$, by Definition 6.21, and, thus, $\mathbf{K}\, A \in T_{\mathcal{K}_G/\mathbf{N}_n} \uparrow \omega$ by Definition 6.14. So $\mathbf{K}\, A \in T_{\mathcal{K}_G/\mathbf{N}_n} \uparrow m$ for some m and we show by induction on m that $\mathbf{K}\, A \in T_{\mathcal{K}^d_G//'\mathbf{N}_n} \uparrow m$ (2), which implies that $\mathbf{K}\, A \in \mathbf{P}^d_{n+1}$. The base case for $m = 0$ holds immediately. Assume the claim (2) holds for m, we show it for $m+1$. Suppose that $\mathbf{K}\, A \in T_{\mathcal{K}_G/\mathbf{N}_n} \uparrow (m+1)$ then $\mathbf{K}\, A \in T_{\mathcal{K}_G/\mathbf{N}_n}(T_{\mathcal{K}_G/\mathbf{N}_n} \uparrow m)$. Then either $\mathbf{K}\, A \in R_{\mathcal{K}_G/\mathbf{N}_n}(T_{\mathcal{K}_G/\mathbf{N}_n} \uparrow m)$ or $\mathbf{K}\, A \in D_{\mathcal{K}_G/\mathbf{N}_n}(T_{\mathcal{K}_G/\mathbf{N}_n} \uparrow m)$. We start with the first case, i.e., there is a rule $\mathbf{K}\, A \leftarrow \mathbf{K}\, A_1, \ldots, \mathbf{K}\, A_n, \mathbf{not}\, B_1, \ldots, \mathbf{not}\, B_m$ with $\mathbf{K}\, A_i \in T_{\mathcal{K}_G/\mathbf{N}_n} \uparrow m$ and $\mathbf{not}\, B_j \notin \mathbf{N}_n$ for all i and j. For each such rule $\mathbf{K}\, A \leftarrow \mathbf{K}\, A_1, \ldots, \mathbf{K}\, A_n, \mathbf{not}\, B_1, \ldots, \mathbf{not}\, B_m$ in $\mathcal{K}_G$ there is, according to Definition 8.1, a rule $\mathbf{K}\, A \leftarrow \mathbf{K}\, A_1, \ldots, \mathbf{K}\, A_n, \mathbf{not}\, B^d_1, \ldots, \mathbf{not}\, B^d_m$ in $\mathcal{P}^d_G$. Since, by the induction hypothesis (1), we have that $\mathbf{K}\, B_i \notin \mathbf{N}_n$ if and only if $\mathbf{K}\, B^d_i \notin \mathbf{N}^d_n$ we obtain that each rule in $\mathcal{K}_G/\mathbf{N}_n$ has its correspondent in $\mathcal{K}^d_G//'\mathbf{N}^d_n$. Then we obtain by the nested induction hypothesis of (2) that $\mathbf{K}\, A \in T_{\mathcal{K}^d_G//'\mathbf{N}_n} \uparrow (m+1)$. Otherwise, $\mathbf{K}\, A \in D_{\mathcal{K}_G/\mathbf{N}_n}(T_{\mathcal{K}_G/\mathbf{N}_n} \uparrow m)$ holds, and $\mathbf{K}\, A \in D_{\mathcal{K}^d_G//'\mathbf{N}_n}(T_{\mathcal{K}^d_G//'\mathbf{N}_n} \uparrow m)$ is obtained immediately by the induction hypothesis (2) and the identical ontologies $\mathcal{O}$ contained in $\mathcal{K}^d_G$ and $\mathcal{K}_G$.

(ii) To prove (ii) we suppose as well that $\mathbf{K}\,B \notin \mathbf{N}_{n+1}$ but $\mathbf{K}\,B \in \mathbf{N}_n$. We show that $\mathbf{K}\,B^d \notin \mathbf{N}^d_{n+1}$. If $\mathbf{K}\,B \notin \mathbf{N}_{n+1}$, then $\mathbf{K}\,B \in U_{\mathcal{K}_G}(\mathbf{P}_n, \mathsf{KA}(\mathcal{K}_G) \setminus \mathbf{N}_n)$. By Definitions 6.30, 8.8, and 8.1, we obtain that $\mathbf{K}\,B \in U_{\mathcal{K}^d_G}(\mathbf{P}^d_n, \mathsf{KA}(\mathcal{K}^d_G) \setminus \mathbf{N}^d_n)$. Hence, by Lemma 8.9, $\mathbf{K}\,B \notin \mathbf{N}^d_{n+1}$.

□

It follows immediately from this proposition that we can use this alternative computation to compute the well-founded MKNF model. Formally, we obtain a theorem that shows the adapted well-founded partition from which the well-founded MKNF model can be obtained.

Theorem 8.11. *Let $\mathcal{K}_G = (\mathcal{O}, \mathcal{P}_G)$ be a ground consistent hybrid MKNF knowledge base and let $\mathbf{P}^d_{\mathcal{K}_G}, \mathbf{N}^d_{\mathcal{K}_G} \subseteq \mathsf{KA}(\mathcal{K}^d_G)$ with $\mathbf{P}^d_{\mathcal{K}_G} = \{\mathbf{K}\,A \mid \mathbf{K}\,A \in \mathbf{P}^d_\omega$ and $\mathbf{K}\,A \in \mathsf{KA}(\mathcal{K}_G)\}$ and $\mathbf{N}^d_{\mathcal{K}_G} = \{\mathbf{K}\,A^d \mid \mathbf{K}\,A^d \in \mathbf{N}^d_\omega$ and $\mathbf{K}\,A \in \mathsf{KA}(\mathcal{K}_G)\}$. Then*

$$M_{WF} = (\{\mathbf{K}\,A \mid \mathbf{K}\,A \in \mathbf{P}^d_{\mathcal{K}_G}\} \cup \{\mathbf{K}\,\pi(\mathcal{O})\}, \{\mathbf{K}\,A \mid \mathbf{K}\,A^d \in \mathsf{KA}(\mathcal{K}^d_G) \setminus \mathbf{N}^d_{\mathcal{K}_G}\})$$

is the well-founded partition *of $\mathcal{K}_G$.*

Proof. The result is an immediate consequence of Proposition 8.10 and Definition 6.25. □

The sets $\mathbf{P}^d_{\mathcal{K}_G}, \mathbf{N}^d_{\mathcal{K}_G}$ are only used to remove modal atoms that are not of interest for the two contexts. We note that for practical purposes we also derive from this theorem and Proposition 8.10 that we have to use the new predicates A^d if we query for negative literals.

We finish the section with a technical example showing the computation and its impact on inconsistency detection.

Example 8.12. *The subsequent example illustrates the explosive behavior of the iteration in the case of an MKNF-inconsistency and that the sets of computed literals for both computations indeed coincide (as proven in Proposition 8.10). Consider the*

knowledge base $\mathcal{K}_G$[1].

$$\mathtt{Q} \sqsubseteq \neg \mathtt{R} \tag{8.1}$$

$$\mathtt{p(a)} \leftarrow \mathbf{not}\, \mathtt{p(a)} \tag{8.2}$$

$$\mathtt{Q(a)} \leftarrow \tag{8.3}$$

$$\mathtt{R(a)} \leftarrow \mathbf{not}\, \mathtt{R(a)} \tag{8.4}$$

We can compute the two sequences $\mathbf{P}_i$ *and* $\mathbf{N}_i$ *and obtain:*

$$\mathbf{P}_0 = \emptyset \qquad \mathbf{N}_0 = \{\mathtt{p(a)}, \mathtt{Q(a)}, \mathtt{R(a)}\}$$
$$\mathbf{P}_1 = \{\mathtt{Q(a)}\} \qquad \mathbf{N}_1 = \mathbf{N}_0$$
$$\mathbf{P}_2 = \mathbf{P}_1 \qquad \mathbf{N}_2 = \{\mathtt{p(a)}, \mathtt{Q(a)}\}$$
$$\mathbf{P}_3 = \{\mathtt{p(a)}, \mathtt{Q(a)}, \mathtt{R(a)}\} \qquad \mathbf{N}_3 = \mathbf{N}_2$$
$$\mathbf{P}_4 = \mathbf{P}_3 \qquad \mathbf{N}_4 = \emptyset$$

The knowledge base is obviously inconsistent since we derive that everything is true and false at the same time.

Now we apply the alternative computation using the doubled set of rules $\mathcal{P}_G^d$ *and the ontology* $\mathcal{O}$ *and its renaming* $\mathcal{O}^d$ *including the special marker predicates* NR *and* NQ*.*

$$\mathtt{Q} \sqsubseteq \neg \mathtt{R} \qquad \mathtt{Q^d} \sqsubseteq \neg \mathtt{R^d}$$
$$\mathtt{p(a)} \leftarrow \mathbf{not}\, \mathtt{p^d(a)} \qquad \mathtt{p^d(a)} \leftarrow \mathbf{not}\, \mathtt{p(a)}$$
$$\mathtt{Q(a)} \leftarrow \qquad \mathtt{Q^d(a)} \leftarrow \mathbf{not}\, \mathtt{NQ(a)}$$
$$\mathtt{R(a)} \leftarrow \mathbf{not}\, \mathtt{R^d(a)} \qquad \mathtt{R^d(a)} \leftarrow \mathbf{not}\, \mathtt{R(a)}, \mathbf{not}\, \mathtt{NR(a)}$$

We compute the two sequences for the transformed knowledge base $\mathcal{K}_G^d$ *and obtain:*

$$\mathbf{P}_0^d = \emptyset \qquad \mathbf{N}_0^d = \{\mathtt{p(a)}, \mathtt{p^d(a)}, \mathtt{Q(a)}, \mathtt{Q^d(a)}, \mathtt{R(a)}, \mathtt{R^d(a)}, \mathtt{NQ(a)}, \mathtt{NR(a)}\}$$
$$\mathbf{P}_1^d = \{\mathtt{Q(a)}, \mathtt{Q^d(a)}\} \qquad \mathbf{N}_1^d = \mathbf{N}_0^d$$
$$\mathbf{P}_2^d = \mathbf{P}_1^d \qquad \mathbf{N}_2^d = \{\mathtt{p(a)}, \mathtt{p^d(a)}, \mathtt{Q(a)}, \mathtt{Q^d(a)}, \mathtt{R(a)}\}$$
$$\mathbf{P}_3^d = \{\mathtt{p(a)}, \mathtt{Q(a)}, \mathtt{R(a)}\} \qquad \mathbf{N}_3^d = \mathbf{N}_2^d$$
$$\mathbf{P}_4^d = \mathbf{P}_3^d \qquad \mathbf{N}_4^d = \{\mathtt{p(a)}, \mathtt{Q(a)}, \mathtt{R(a)}\}$$

All atoms based on original predicates are true while all doubled atoms are false. This indicates again that the knowledge base is MKNF-inconsistent. In both sequences,

[1]Note that the modal operator **K** does not appear in the example. To simplify notation here and in the remaining chapters, we assume that the modal operator **K** is implicit in rules of hybrid MKNF knowledge bases, while **not** is maintained to represent default negation.

the doubled predicates appear along with the original ones; only the inconsistent case of $\mathtt{R}(\mathtt{a})$ *is different. In the increasing sequence* $\mathbf{P}_i$*, only* $\mathtt{R}(\mathtt{a})$ *is added, while in the decreasing sequence* $\mathbf{N}_i$ *only* $\mathtt{R}^{\mathtt{d}}(\mathtt{a})$ *is removed. The detection of inconsistencies is thus straightforward by means of the alternative computation.*

Note that the inconsistency in $\mathtt{R}$ ensures that everything in the knowledge base is considered inconsistent. This does not always hold (consider adding a fact $\mathtt{p}(\mathtt{a}) \leftarrow$ to the rules, then $\mathtt{p}^{\mathtt{d}}(\mathtt{a})$ is not false but true). However, whenever we encounter an atom such that $\mathtt{P}$ is true while $\mathtt{P}^{\mathtt{d}}$ is false then we know that the KB is inconsistent. This is not a sufficient condition but we do not consider that a problem. We want to query a knowledge base and only consider the information relevant for the query and not the whole knowledge base. So, we cannot always detect such possibly occurring inconsistencies.

8.2 Top-Down Queries with SLGO

Now, we present the new top-down procedure **SLG($\mathcal{O}$)** for Hybrid MKNF knowledge bases, which extends SLG resolution from [Chen and Warren, 1996] with an oracle to capture first-order deduction using a description logic. SLG is a form of tabled resolution that handles loops within the program, and does not change the data complexity of WFS. It does that by resorting to already computed results in a forest of derivation trees, a technique also known as *tabling*. If SLG is extended with an oracle, then many of the definitions of SLG are affected; in this section we recall the definitions of SLG, and at the same time show how the definitions are extended to communicate with an oracle, as well as defining when an oracle is suitable for use in an evaluation.

SLG evaluations are sequences of forests (sets) of program trees.[1] Program trees themselves correspond to subgoals[2] that have been encountered in an evaluation. The nodes in these trees contain sets of literals divided into those literals that have not been examined, and others that have been examined, but their resolution delayed (cf. Definition 8.15). The need to delay some literals arises for the following reason. Modern Prolog engines rely on a fixed order for selecting literals in a rule, e.g., always

[1] Here, we model SLG derivations using the forest of trees model of [Swift, 1999].

[2] Here and in the rest of the thesis, the term subgoal is used in a broad sense that may refer to one element of a conjunctive query, but also to a subsequent goal in a derivation process.

left-to-right. However, well-founded computations cannot be performed using a fixed-order literal selection function.[1] Hence, in SLG some literals may be delayed and later resolved through an operation called SIMPLIFICATION. In addition to modeling the operational behavior of Prolog, the use of delay and SIMPLIFICATION supports the termination and complexity results that are discussed in Section 8.3.

Example 8.13. *We present SLG resolution on an intuitive level using logic program P from Example 2.29.*

$$\mathtt{a} \leftarrow \tag{8.5}$$

$$\mathtt{b} \leftarrow \mathbf{not}\ \mathtt{a}, \mathtt{c} \tag{8.6}$$

$$\mathtt{c} \leftarrow \mathbf{not}\ \mathtt{b} \tag{8.7}$$

$$\mathtt{d} \leftarrow \mathtt{a}, \mathbf{not}\ \mathtt{d} \tag{8.8}$$

$$\mathtt{e} \leftarrow \mathbf{not}\ \mathtt{c}, \mathtt{f} \tag{8.9}$$

$$\mathtt{f} \leftarrow \mathtt{e} \tag{8.10}$$

Consider that we query for d*. We start with a tree with root* d *and check for rules whose heads are unifiable (here simply identical) with the root. The forth rule matches and we may add a child* a, **not** d *corresponding to the body of the rule to the root* d*. We continue evaluating that child and encounter* a*. We create a new tree with root* a *that can be immediately resolved with the first rule upon which we obtain an empty child. Thus,* a *is true and we can create a child* **not** d *for* a, **not** d*. In general, we may now create a new tree with root d but since that tree is already existing we are not allowed to proceed like that. Instead, we can only delay* **not** d *creating the conditional answer* d : −**not** d | *that corresponds to the expectation that* d *should be undefined. Maintaining all the already obtained trees, we may query for* b*. We create a new tree with root* b *and a child* **not** a, c *according to the second rule. Then, we can use that* a *is already evaluated to true and evaluate that child with* **not** a *to false. Since it is the only possible child for* b*,* b *must be false as expected.*

Next, we formalize this intuitive presentation from the example and extend it with an oracle to the DL starting with the definition of delay literals.

[1] A literal selection function is employed to choose the next literal to resolve in set of goals/body of a rule. In **SLG($\mathcal{O}$)**, the only requirement for a selection function is that DL-atoms are not selected until they are ground, which is always possible given DL-safety of conjunctive queries and the rules appearing in the knowledge base.

Definition 8.14. *A* negative delay literal *has the form* **not** *A, where A is a ground atom.* Positive delay literals *have the form* A^{Call}_{Answer}*, where A is an atom whose truth value depends on the truth value of some literal Answer for the literal Call. If* θ *is a substitution, then* $(A^{Call}_{Answer})\theta = (A\theta)^{Call}_{Answer}$.

Positive delay literals can only appear as a result of resolution (see Definition 8.16 below). Their special form, as indicated by the indices, is meant to keep track of the answer and call used for that resolution, and possible substitutions are thus only applied to A itself.

A derivation proceeds by constructing a forest according to the set of operations in Definition 8.21 based on the rules and the ontology of the knowledge base. Such a forest, and the trees and nodes it contains are defined as follows.

Definition 8.15. *A* node *has the form*

$$AnswerTemplate \text{ :- } Delays|Goals \qquad or \qquad \text{fail}.$$

In the first form, AnswerTemplate is an atom or a classically negated atom, Delays is a sequence of (positive and negative) delay literals and Goals is a sequence of literals. The second form is called a failure node. *A* program tree T *is a tree of nodes whose root is of the form* S :- $|S$ *for some atom* S*: we call* S the root node for T *and* T the tree for S. *An* SLG forest $\mathcal{F}$ *is a set of program trees. A node* N *is an* answer *when it is a leaf node for which Goals is empty. If Delays of an answer is empty it is termed an* unconditional answer*, otherwise, it is a conditional answer. Program trees* T *may be marked as complete.*

This definition is almost identical to SLG resolution. The only difference is that we also allow for the appearance of classically negated atoms as roots to incorporate possible calls for the classical negation as required by the bottom-up computation in Definition 8.3. Such a literal $\neg A$ does only appear in AnswerTemplate and as the only goal in the root node, and may only be used to query $\mathcal{O}$.

The definition of answer resolution in $\mathbf{SLG}(\mathcal{O})$ is slightly different from the usual one to take into account delay literals in conditional answers.

Definition 8.16. *Let* N *be a node* A :- $D|L_1, ..., L_n$*, where* $n > 0$*. Let* $Ans = A'$:- $D'|$ *be an answer whose variables are disjoint from* N*.* N *is* SLG resolvable *with Ans if*

$\exists i$, $1 \leq i \leq n$, *such that* L_i *and* A' *are unifiable with an mgu*[1] θ. *The* SLG resolvent *of* N *and Ans on* L_i *has the form:*

$$(A \text{ :- } D | L_1, ..., L_{i-1}, L_{i+1}, ..., L_n)\theta$$

if D' *is empty; otherwise the resolvent has the form:*

$$(A \text{ :- } D, L_i{}^{L_i}_{A'} | L_1, ..., L_{i-1}, L_{i+1}, ..., L_n)\theta$$

Note that we delay L_i rather than propagating the answer's delay list. This is necessary, as shown in [Chen and Warren, 1996], to ensure polynomial data complexity.[2]

At certain points in **SLG($\mathcal{O}$)** resolution, a set of goals may be completely evaluated, i.e., they are either already proven true, or no further expansion by means of the knowledge base is possible. For that purpose also the notion of an underlying subgoal is necessary.

Definition 8.17. *The* underlying subgoal *of (an extended) literal* L *is* L *if* L *is positive or* $L = \neg S$, *it is* S *if* $L = \textbf{not}\ S$ *(and* S *is not based on one of the new predicates* NH *introduced in Definition 8.3), or it is* $\neg H(t_i)$ *if* $L = \textbf{not}\ NH(t_i)$. *A set* $\mathcal{S}$ *of literals is* completely evaluated *if at least one of the conditions holds for each* $S \in \mathcal{S}$

1. *The tree for* S *contains an answer* S *:- |; or*
2. *For each node* N *in the tree for* S*:*
   (a) *The underlying subgoal of the selected literal of* N *is completed; or*
   (b) *The underlying subgoal of the selected literal of* N *is in* $\mathcal{S}$ *and there are no applicable* New Subgoal, Program Clause Resolution, Oracle Resolution, Positive Return, Negative Return *or* Delaying *operations (Definition 8.21) for* N.

Once a set of literals is determined to be completely evaluated, the Completion operation marks the trees for each literal (Definition 8.15). If a subgoal S is completed due to condition 1 holding, we say that S is *early completed*. If condition 1 does not hold, condition 2 of the above definition prevents the Completion operation from being applied to a set of trees if certain other operations are still applicable to those

[1] most general unifier

[2] If delay lists were propagated directly, then delay lists could contain all derivations which could be exponentially many in bad cases.

trees.[1] This notion of completion is incremental in the sense that once a set $\mathcal{S}$ of mutually dependent subgoals is fully evaluated, the derivation does not need to be concerned with the trees for $\mathcal{S}$ apart from any answers they contain. In an actual implementation resources for such trees can be reclaimed.

In certain cases the propagation of conditional answers through resolution (Definition 8.16) can lead to a set of *unsupported answers* — conditional answers that are false in the well founded model (see, e.g., Example 1 of [Swift et al., 2009]).[2] Intuitively, these answers, which are mutually dependent, correspond to an unfounded set, but their technical definition is based on the form of conditional answers.

Definition 8.18. *Let $\mathcal{F}$ be an SLG forest, S a root of a tree in $\mathcal{F}$, and Answer be an atom that occurs in the head of some answer of S. Then Answer is* supported *by S in $\mathcal{F}$ if and only if:*

1. *S is not completely evaluated; or*
2. *there exists an answer node Answer :- Delays| of S such that for every positive delay literal D^{Call}_{Ans}, Ans is supported by Call.*

As mentioned in Definition 8.15, the root of each tree in an SLG forest corresponds to a subgoal. Expanding on this correspondence, we may associate an SLG forest with a partial interpretation, taking into consideration that, besides atoms and default negated atoms as in SLG, we also allow the appearance of classical negated roots. This interpretation is shown to correspond to M_{WF} (cf. Theorem 8.26 below).

Definition 8.19. *Let $\mathcal{F}$ be a forest. Then the* interpretation induced by *$\mathcal{F}$, $I_{\mathcal{F}}$, is the smallest set such that:*

- *A (ground) atom $A \in I_{\mathcal{F}}$ iff A is in the ground instantiation of some unconditional answer Ans :- | in $\mathcal{F}$.*
- *A (ground) negated atom $\neg A \in I_{\mathcal{F}}$ iff $\neg A$ is in the ground instantiation of some unconditional answer Ans :- | in $\mathcal{F}$.*

[1] "Completely evaluated" is thus a condition that is used for prioritizing operations, which is why its definition contains a forward reference to Definition 8.21.

[2] As an aside, we note that unsupported answers appear to be uncommon in practical evaluations which minimize the use of delay such as[Sagonas et al., 2000].

- *A (ground) literal* **not** $A \in I_{\mathcal{F}}$ *iff* A *is in the ground instantiation of a completely evaluated literal in* $\mathcal{F}$*, and* A *is not in the ground instantiation of any answer in a tree in* $\mathcal{F}$*.*

An atom S *is* successful *(resp.* failed*) in* $I_{\mathcal{F}}$ *if* S' *(resp.* **not** S'*) is in* $I_{\mathcal{F}}$ *for every* S' *in the ground instantiation of* S*. A negative delay literal* **not** D *is successful (resp. failed) in a forest* $\mathcal{F}$ *if* D *is failed (resp. successful) in* $\mathcal{F}$*. Similarly, a positive delay literal* D^{Call}_{Ans} *is successful (failed) in a* $\mathcal{F}$ *if* $Call$ *has an unconditional answer* Ans *:- | in* $\mathcal{F}$*.*

This definition extends the corresponding one from SLG by allowing the occurrence of classically negated literals in $I_{\mathcal{F}}$.

The next step in describing **SLG($\mathcal{O}$)** is to characterize the behavior of an abstract oracle, $\mathcal{O}$,[1] that computes entailment according to the DL knowledge base $\mathcal{O}$. For that purpose, we define an oracle transition function that in a single step computes all possible atoms required to prove the goal. In other words, such an oracle, when posed a query S, non-deterministically returns in one step a set of atoms L defined in the program, i.e., atoms for which there is at least one rule with it in the head, such that, if added to the oracle theory, immediately entail S. We only have to take into account that we query appropriately $\mathcal{O}$ or its renaming $\mathcal{O}^d$ (see Section 8.1), and that we only add the positive part of $I_{\mathcal{F}}$.

Definition 8.20. *Let* $\mathcal{K}^d = (\mathcal{O}, \mathcal{O}^d, \mathcal{P}^d)$ *be a doubled hybrid MKNF knowledge base,* S *a goal,* L *a set of ground atoms which appear in at least one rule head in* $\mathcal{P}^d_G$*, and* $I^+_{\mathcal{F}_n} = I_{\mathcal{F}_n} \setminus \{\textbf{not}\ A \mid \textbf{not}\ A \in I_{\mathcal{F}_n}\}$*. The* complete oracle *for* $\mathcal{O}$*, denoted* $compT_{\mathcal{O}}$*, is defined by*

$$compT_{\mathcal{O}}(I_{\mathcal{F}_n}, S, L) \text{ iff } \mathcal{O} \cup I^+_{\mathcal{F}_n} \cup L \models S \text{ or } \mathcal{O}^d \cup I^+_{\mathcal{F}_n} \cup L \models S$$

The set $\mathcal{O} \cup I^+_{\mathcal{F}_n} \cup L$ (and likewise $\mathcal{O}^d \cup I^+_{\mathcal{F}_n} \cup L$) may be inconsistent even though $\mathcal{K}$ is MKNF-consistent. Consequently, such a complete oracle potentially allows us to obtain a large number of entailments that are eventually useless to derive S if $\mathcal{K}$ is MKNF-consistent. Later in this section, we provide a partial oracle that overcomes this lack of efficiency.

[1] We overload $\mathcal{O}$ syntactically to represent the oracle and the ontology, i.e., its underlying DL knowledge base, since from the viewpoint of SLG_O they are the same.

As already pointed out, in a practical system we can avoid maintaining two renamings of the ontology by appropriately transmitting the contents of $I_{\mathcal{F}_n}$. For example, when querying for an atom A^d we can simply transmit all atoms based on doubled predicates in its original form. The set L is then transformed into its doubled correspondent.

We are now able to characterize the SLG($\mathcal{O}$) operations that build the various trees of the evaluation forest.

Definition 8.21 (SLG($\mathcal{O}$) Operations)**.** *Let $\mathcal{K}^d = (\mathcal{O}, \mathcal{O}^d, \mathcal{P})$ be a doubled hybrid MKNF knowledge base. Given a forest $\mathcal{F}_n$ of an SLG($\mathcal{O}$) evaluation of $\mathcal{K}^d$, $\mathcal{F}_{n+1}$ may be produced by one of the following operations.*

1. New Subgoal*: Let $\mathcal{F}_n$ contain a tree with non-root node*
$$N = Ans \text{ :- } Delays | G, Goals$$
where G is the selected literal S or **not** *S (and S is not based on one of the new predicates NH introduced in Definition 8.3). Alternatively, $G =$* **not** *$NH(t_i)$ and $S = \neg H(t_i)$. Assume $\mathcal{F}_n$ contains no tree with root S. Then add the tree S :- $|S$ to $\mathcal{F}_n$.*

2. Program Clause Resolution*: Let $\mathcal{F}_n$ contain a tree with root node $N = S$:- $|S$ and C be a rule $Head$:- $Body$ such that $Head$ unifies with S with mgu θ. Assume that in $\mathcal{F}_n$, N does not have a child $N_{child} = (S$:- $|Body)\theta$. Then add N_{child} as a child of N.*

3. Oracle Resolution*: Let $\mathcal{F}_n$ contain a tree with root node $N = S$:- $|S$. Assume that $compT_{\mathcal{O}}(I_{\mathcal{F}_n}, S, Goals)$. If N does not have a child $N_{child} = S$:- $|Goals$ in $\mathcal{F}_n$ then add N_{child} as a child of N.*

4. Positive Return*: Let $\mathcal{F}_n$ contain a tree with non-root node N whose selected literal S is positive. Let Ans be an answer for S in $\mathcal{F}_n$ and N_{child} be the SLG resolvent of N and Ans on S. Assume that in $\mathcal{F}_n$, N does not have a child N_{child}. Then add N_{child} as a child of N.*

5. Negative Return*: Let $\mathcal{F}_n$ contain a tree with a leaf node, whose selected literal* **not** *S is ground*
$$N = Ans \text{ :- } Delays | \textbf{not}\ S, Goals.$$
   (a) Negation Success*: If S is failed in $\mathcal{F}$ then create a child for N of the form: Ans :- $Delays|Goals$.*

(b) NEGATION FAILURE: *If S succeeds in $\mathcal{F}$, then create a child for N of the form* fail.

6. DELAYING: *Let $\mathcal{F}_n$ contain a tree with leaf node $N = Ans$:- $Delays|\mathbf{not}\, S, Goals$, such that S is ground in $\mathcal{F}_n$, but S is neither successful nor failed in $\mathcal{F}_n$. Then create a child for N of the form Ans :- $Delays, \mathbf{not}\, S|Goals$.*

7. SIMPLIFICATION: *Let $\mathcal{F}_n$ contain a tree with leaf node $N = Ans$:- $Delays|$, and let $L \in Delays$*

   (a) *If L is failed in $\mathcal{F}$ then create a child* fail *for N.*

   (b) *If L is successful in $\mathcal{F}$, then create a child Ans :- $Delays'|$ for N, where $Delays' = Delays - L$.*

8. COMPLETION: *Given a completely evaluated set $\mathcal{S}$ of literals (Definition 8.17), mark the trees for all literals in $\mathcal{S}$ as completed.*

9. ANSWER COMPLETION: *Given a set of unsupported answers $\mathcal{UA}$, create a failure node as a child for each answer $Ans \in \mathcal{UA}$.*

The operation NEW SUBGOAL creates a new tree in the forest $\mathcal{F}$ for a literal appearing in the set of goals in some (non-root) node in a tree in $\mathcal{F}$, handling also the case of calls for the classical negation.[1] Two different operations apply to the root node of a tree for a subgoal S. PROGRAM CLAUSE RESOLUTION resolves a root node against rules in the knowledge base, while ORACLE RESOLUTION queries the ontology to obtain atoms that, if added to $\mathcal{O}$, entail S. Of course, for non-DL atoms this latter operation is limited to derivations based on equality. A third set applies to non-root nodes. POSITIVE RETURN resolves positive literals in nodes of trees with answers according to Definition 8.16, while NEGATIVE RETURN handles the negative literals: if the literal itself fails then the negated literal is simply removed (NEGATION SUCCESS), otherwise if the literal itself is successful, then the whole branch is failed (NEGATION FAILURE). DELAYING allows to delay negative literals, while SIMPLIFICATION allows to simplify positive and negative delay literals. Finally, COMPLETION allows completion of trees while ANSWER COMPLETION removes unsupported answers in accordance with Definition 8.18. The only differences to SLG resolution are the operation ORACLE

[1] Note that the corresponding tree is always created even if $\neg H(t_i)$ is not expressible. In this case simply no ORACLE RESOLUTION is applicable and the tree eventually fails.

RESOLUTION and the extension of NEW SUBGOAL to incorporate calls for classically negated atoms. Thus each operation of Definition 8.21 can be seen as a function that associates a forest with a new forest by adding a new tree, adding a new node to an existing tree, or marking a set of trees as complete. The only thing missing to complete the description of the procedure is the formalization of the initialization of an SLG evaluation, i.e., how the initial (DL-safe) conjunctive query is defined.

Definition 8.22. *Let $\mathcal{K}^d$ be a doubled hybrid MKNF knowledge base and let q be a query of the form $q(X_i) \leftarrow A_1, \ldots, A_n, \mathbf{not}\, B_1^d, \ldots, \mathbf{not}\, B_m^d$ where X_i is the (possibly empty) set of requested variables. We set $\mathcal{F}_0 = \{q(X_i) : - \mid q(X_i)\}$ to be the initial forest of an SLG($\mathcal{O}$) evaluation of $\mathcal{K}^d$ for q.*

Of course, if the query is atomic we can simply start with that atomic query, i.e., with the root containing the queried literal itself. Since the derivation uses $\mathcal{K}^d$ (the doubled knowledge base), the technically correct way to query negative literals is to use $\mathbf{not}\, B^d$ instead of $\mathbf{not}\, B$ for any atom B which is why we use the doubled predicates for negative literals in the query.

In the next section, we show that **SLG($\mathcal{O}$)** always terminates (Theorem 8.24) and, even though some orders of application of the possible operations are more efficient than others, that the procedure is confluent (Theorem 8.25). We also show that the procedure is sound and complete w.r.t. the well-founded MKNF model (Theorem 8.26) and that it is sound w.r.t. the semantics of two-valued MKNF (Corollary 8.27). Finally, under some assumptions, we maintain the computational complexity of the bottom-up procedure (Theorem 8.30), which is actually an improvement since we do not have to consider the entire knowledge base but only the part relevant for a concrete query.

But before we come to showing these results and the respective proofs, we finish the presentation of the procedure **SLG($\mathcal{O}$)** with an example illustrating its behavior.

Example 8.23. *In order to illustrate the actions of SLG($\mathcal{O}$) we consider a derivation of an answer to the query* `?- discount(Bill)` *using a KB $\mathcal{K}$ from [Motik and Rosati, 2007]*[1]*:*

[1]For ease of reading and since neither an MKNF-inconsistency nor an issue related to coherence occurs, we operate on $\mathcal{K}$ directly instead of on $\mathcal{K}_G^d$.

$$\texttt{NonMarried} \equiv \neg\texttt{Married} \tag{8.11}$$

$$\neg\texttt{Married} \sqsubseteq \texttt{HighRisk} \tag{8.12}$$

$$\exists\texttt{Spouse.T} \sqsubseteq \texttt{Married} \tag{8.13}$$

$$(\exists\texttt{Spouse.}\{\texttt{Michelle}\})(\texttt{Bill}) \tag{8.14}$$

$$\texttt{NonMarried(x)} \leftarrow \textbf{not}\ \texttt{Married(x)} \tag{8.15}$$

$$\texttt{discount(x)} \leftarrow \textbf{not}\ \texttt{HighRisk(x)} \tag{8.16}$$

First, note that TBox and ABox information are each distributed over both the description logic and the rules. Figure 8.1 shows the final forest for this evaluation, where elements are marked in the order they are created. The initial forest for the evaluation consists of node 0 only. Since the selected literal of node 0, discount(Bill) *is a non-DL-atom and there are no equalities in the KB, we can only apply* Program Clause Resolution *with (8.16), which produces node 1, followed by* New Subgoal *to produce node 2. Node 2 is a DL-atom with no rules applicable for* HighRisk(Bill)*, but an* Oracle Resolution *operation can be applied to derive from (8.11) and (8.12) that if* NonMarried(Bill) *can be proven (node 3), then this suffices to prove* HighRisk(Bill)*. Then, via a* New Subgoal *operation, node 4 is obtained. The selected literal for node 4,* NonMarried(Bill)*, is a DL-atom that also is the head of a rule, so the oracle and the program evaluation are both applicable to derive the atom. On the program side,* Program Clause Resolution *produces node 5 from (8.15) and* New Subgoal *produces node 6. The selected literal of node 6,* Married(Bill)*, is a DL-atom that is not the head of a program rule, so once again the only possibility is to use* Oracle Resolution*, and the answer* Married(Bill) *is derived from (8.13) and (8.14). Using this answer, the tree for* Married(Bill) *can be early completed and a* Negative Return *operation produces node 8. The tree for* NonMarried(Bill)*, which does not have an answer, must be completed (step 10), and the same holds for* HighRisk(Bill) *(step 11). Once this occurs, a* Negative Return *operation is enabled to produce node 12.*

The evaluation illustrates several points. First, the evaluation makes use of classical negation in the ontology along with closed world negation in the rules. From an operational perspective, the actions of the description logic prover and the program are interleaved, with the program "calling" the oracle by creating new trees for DL-atoms, and the oracle "calling" the rule system through Oracle Resolution *operations. As a result, trees for DL-atoms must either be early-completed, or explicitly completed by the tabulation system.*

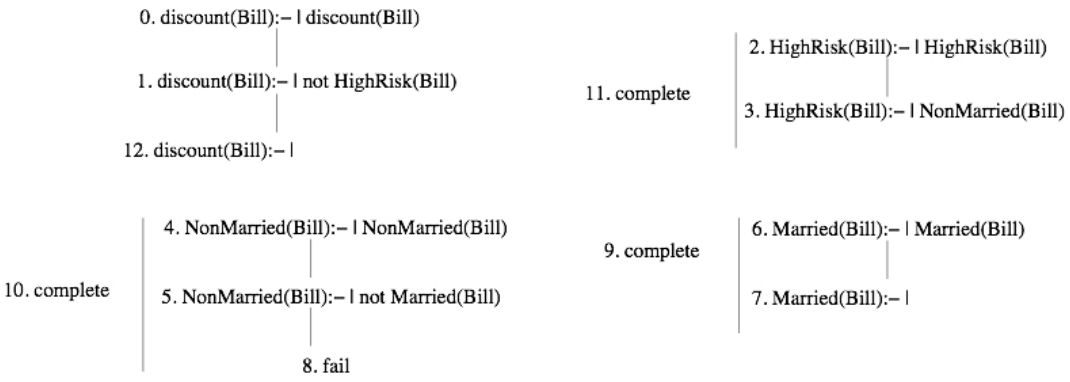

Figure 8.1: Final Forest for the query `discount(Bill)` to $\mathcal{K}$

8.3 Properties of SLG($\mathcal{O}$)

We now present and prove several important properties of **SLG($\mathcal{O}$)**-resolution. The first property we can ensure is that our extension of SLG resolution terminates for the evaluation of any query, generating a final forest.

Theorem 8.24. *Let $q = L$ be a query to a doubled, hybrid MKNF knowledge base $\mathcal{K}^d$. Then any* **SLG($\mathcal{O}$)** *evaluation of q terminates after finitely many steps, producing a finite final forest.*

Proof. The proof is straightforward since we know already that SLG, i.e., **SLG($\mathcal{O}$)** without Oracle Resolution and the extended New Subgoal operation, terminates finitely for programs with bounded term-depth, and transfinitely otherwise (cf. Theorem 5.10 of [Chen and Warren, 1996]). Since Definition 4.16 ensures that hybrid MKNF knowledge bases do not contain recursive terms, i.e., non-nullary functors, they have bounded term depth, and so does the doubled knowledge base $\mathcal{K}^d$. Accordingly, we only have to ensure that the new operation Oracle Resolution and the extension of New Subgoal do not invalidate finite termination.

The operation Oracle Resolution can be applied in the same situation as Program Clause Resolution, namely when creating a new child for a root of a tree, so that each operation can be applied only once to a given node (for each of the finitely many rules, respectively for each of the finitely many possible answers of the complete oracle), and this creates one child per successful application. Now, since the knowledge base $\mathcal{K}^d$ is finite, the number of (ground) rule heads is finite. Thus, 1) the number of children possibly created with Oracle Resolution for any arbitrary root is finite; and 2) the size of the nodes created is also finite.

The extension of the operation NEW SUBGOAL creates even in the worst case finitely many more trees with roots to which only ORACLE RESOLUTION is applicable, which does only create finitely many nodes.

We conclude that termination holds for **SLG($\mathcal{O}$)**. □

In the way **SLG($\mathcal{O}$)** is defined, there is no prescribed order in which to apply the operations possible in a forest $\mathcal{F}_i$. For SLG some orders of application are in general more efficient than others but, as shown in [Chen and Warren, 1996], any order yields the same outcome for any query. The same sort of confluence also holds for **SLG($\mathcal{O}$)**:

Theorem 8.25. *Let $\mathcal{E}_1$ and $\mathcal{E}_2$ be two* **SLG($\mathcal{O}$)** *evaluations of a query $q = L$ to a doubled, hybrid knowledge base $\mathcal{K}_G^d$, $\mathcal{F}_1$ the final forest of $\mathcal{E}_1$, and $\mathcal{F}_2$ the final forest of $\mathcal{E}_2$. Then, $I_{\mathcal{F}_1} = I_{\mathcal{F}_2}$.*

Proof. This is a well-known property for SLG as defined using the operations of Definition 8.21 excluding ORACLE RESOLUTION and the extension of NEW SUBGOAL to classical negation (cf. Theorem 5.7 of [Chen and Warren, 1996]). Accordingly, we consider cases in which $\mathcal{E}_1$ and $\mathcal{E}_2$ make use of the operations that have been introduced/extended in **SLG($\mathcal{O}$)**. However, PROGRAM CLAUSE RESOLUTION is used in SLG, and if we just consider the created children, then PROGRAM CLAUSE RESOLUTION and ORACLE RESOLUTION are not distinguishable. Thus, we can consider that ORACLE RESOLUTION is a syntactic variant of PROGRAM CLAUSE RESOLUTION. The same holds for NEW SUBGOAL and the treatment of default negated atoms **not** S that create a tree with root S and those special literals **not** $NH(t_i)$ that may allow us to create a tree with root $\neg H(t_i)$: both its children are not distinguishable and only one of the two is applicable in each case. Thus, confluence of **SLG($\mathcal{O}$)** follows directly from confluence of SLG (see Theorem 5.7 of [Chen and Warren, 1996]). □

The above theorem is also helpful to prove that **SLG($\mathcal{O}$)** is a correct query procedure for the well-founded MKNF model and that it terminates within the same complexity bounds. First, we show that **SLG($\mathcal{O}$)** coincides with the well-founded MKNF model, more precisely with the well-founded partition, from which the well-founded MKNF model is obtained. Intuitively, we have to show that the well-founded MKNF model as presented in Part II and based on the computation presented in Section 8.1, coincides with the interpretation $I_{\mathcal{F}_n}$ induced by $\mathcal{F}_n$ for some query q to $\mathcal{K}^d$ in all the ground literals involved in the query. We can simplify that by showing, for each literal

L appearing in $\mathcal{K}_G^d$, that $L \in M_{WF}$[1] if and only if $L \in I_\mathcal{F}$ with query $q = L$ and $\mathcal{F}_n$ for some n. Note that this correspondence also holds for atoms and classically negated atoms only appearing in the ontology.

Theorem 8.26. *Let $\mathcal{K}_G^d$ be a doubled, ground hybrid MKNF knowledge base, and $I_\mathcal{F}$ the interpretation induced by the forest $\mathcal{F}$ of an* **SLG($\mathcal{O}$)** *evaluation of $\mathcal{K}_G^d$ for query $q = L$ where L is a literal or a classically negated atom.* **SLG($\mathcal{O}$)** *resolution is sound and complete w.r.t. the well-founded partition $M_{WF} = (T_W, F_W)$, which is obtained from $\mathbf{P}_\omega^d$ and $\mathbf{N}_\omega^d$, i.e., we have*

- *for $L \in \mathsf{KA}(\mathcal{K}_G^d)$:*
  - *$L \in \mathbf{P}_\omega^d$ if and only if $L \in I_\mathcal{F}$ and*
  - *$L_1^d \notin \mathbf{N}_\omega^d$ if and only if $L = \mathbf{not}\, L_1^d \in I_\mathcal{F}$.*
- *for $L \notin \mathsf{KA}(\mathcal{K}_G^d)$: $\mathcal{O} \cup \mathbf{P}_\omega^d \models L$ or $\mathcal{O}^d \cup \mathbf{P}_\omega^d \models L$ if and only if $L \in I_\mathcal{F}$.*

Proof. **(Completeness):** We show by induction on n that, for $L \in \mathsf{KA}(\mathcal{K}_G^d)$, if L is a positive literal, then $L \in \mathbf{P}_n^d$ implies that $L \in I_\mathcal{F}$, and if $L = \mathbf{not}\, L_1^d$ is a negative literal, then $L_1^d \notin \mathbf{N}_n^d$ implies that $\mathbf{not}\, L_1^d \in I_\mathcal{F}$, and, for $L \notin \mathsf{KA}(\mathcal{K}_G^d)$, if $\mathcal{O} \cup \mathbf{P}_n^d \models L$ or $\mathcal{O}^d \cup \mathbf{P}_n^d \models L$ then $L \in I_\mathcal{F}$.

The induction base holds immediately, for $L \in \mathsf{KA}(\mathcal{K}_G^d)$, since $\mathbf{P}_0^d$ is empty and $\mathbf{N}_0^d$ contains all literals appearing in $\mathsf{KA}(\mathcal{K}_G^d)$. For $L \notin \mathsf{KA}(\mathcal{K}_G^d)$, we obtain that $\mathcal{O} \models L$ or $\mathcal{O}^d \models L$, so we can create a tree $L : - \mid L$ and with Oracle Resolution an answer $L : - \mid$, which shows $L \in I_\mathcal{F}$.

(1) Now suppose the claim holds for n. We have to show the induction step for $n + 1$. For $L \in \mathsf{KA}(\mathcal{K}_G^d)$, let L be a positive literal, and suppose that $L \in \mathbf{P}_{n+1}^d$ but $L \notin \mathbf{P}_n^d$ (otherwise the claim would immediately follow by the induction hypothesis). Therefore, $L \in \Gamma_{\mathcal{K}_G^d}(\mathbf{N}_n^d)$ and so $L \in T_{\mathcal{K}_G^d//'\mathbf{N}_n^d} \uparrow \omega$. We show by induction on m that $L \in T_{\mathcal{K}_G^d//'\mathbf{N}_n^d} \uparrow m$ implies that $L \in I_\mathcal{F}$.

The base case is void since $T_{\mathcal{K}_G^d//'\mathbf{N}_n^d} \uparrow 0$ is empty. (2) Suppose the property holds for m, we show it for $m + 1$. So, assume that $L \in T_{\mathcal{K}_G^d//'\mathbf{N}_n^d} \uparrow (m + 1)$ but $L \notin T_{\mathcal{K}_G^d//'\mathbf{N}_n^d} \uparrow m$ (otherwise the property would immediately follow by the induction hypothesis (2)). Then $L \in T_{\mathcal{K}_G^d//'\mathbf{N}_n^d}(T_{\mathcal{K}_G^d//'\mathbf{N}_n^d} \uparrow m)$ and either $L \in R_{\mathcal{K}_G^d//'\mathbf{N}_n^d}(T_{\mathcal{K}_G^d/\mathbf{N}_n^d} \uparrow m)$ or $L \in D_{\mathcal{K}_G^d//'\mathbf{N}_n^d}(T_{\mathcal{K}_G^d//'\mathbf{N}_n^d} \uparrow m)$. In the first case, $\mathcal{K}_G^d//'\mathbf{N}_n^d$ contains a rule of the form $L \leftarrow A_1, \ldots, A_n$ such that all $\mathbf{K}\, A_i \in T_{\mathcal{K}_G^d/\mathbf{N}_n^d} \uparrow m$. Additionally, there is a rule $L \leftarrow A_1, \ldots, A_n, \mathbf{not}\, B_1, \ldots, \mathbf{not}\, B_m$ in $\mathcal{K}_G^d$ and all $B_j \notin \mathbf{N}_n^d$. We thus know by

[1] Note that M_{WF} strictly speaking contains $\mathbf{K}$-atoms, but we continue to keep the $\mathbf{K}$ implicit.

the two induction hypotheses that all A_i and all **not** B_j appear in $I_{\mathcal{F}}$. From that we can construct a tree with root $L : - \mid L$ and a child obtained by applying PROGRAM CLAUSE RESOLUTION with the rule considered. In the resulting child the set of goals contains exactly all A_i which can be removed by POSITIVE RETURN and all **not** B_j which can be removed by NEGATIVE RETURN. The result is a leaf node $L : - \mid$ and we obtain that $L \in I_{\mathcal{F}}$ for this order of application. Since Theorem 8.25 ensures that we achieve the same result if we alter the order of applications, we know that the statement holds in general. In the second case, i.e., for $L \in D_{\mathcal{K}_G^d/\mathbf{N}_n^d}(T_{\mathcal{K}_G^d/\mathbf{N}_n^d} \uparrow m)$, we construct a tree $L : - \mid L$ and apply ORACLE RESOLUTION as (finitely) many times as necessary. One of those children is the one actually allowing to derive L by means of the ontology, i.e., all goals in this child are positive literals that are true in $T_{\mathcal{K}_G^d/\mathbf{N}_n^d} \uparrow m$. We apply POSITIVE RETURN to these literals, and this, by the induction hypothesis (2), results in a leaf node $L : - \mid$. As before, Theorem 8.25 ensures that a different application order again yields eventually the same result.

Now, let L be a negative literal **not** L_1^d, and suppose that $L_1^d \notin \mathbf{N}_{n+1}^d$ but $L_1^d \in \mathbf{N}_n^d$ (otherwise the claim would follow immediately by the induction hypothesis (1)). Then, $L_1^d \in U_{\mathcal{K}_G^d}(\mathbf{P}_n^d, \mathsf{KA}(\mathcal{K}_G^d) \setminus \mathbf{N}_n^d)$ by Lemma 8.9, i.e., L_1^d occurs in the greatest unfounded set w.r.t. $(\mathbf{P}_n^d, \mathsf{KA}(\mathcal{K}_G^d) \setminus \mathbf{N}_n^d)$. We construct a tree with root $L_1^d : - \mid L_1^d$. We proceed by creating all children of that root, applying PROGRAM CLAUSE RESOLUTION and ORACLE RESOLUTION as (finitely) many times as possible. By Definition 8.8, each such child (after finitely many subsequent operations) is either false or completely evaluated as in 2.(b) of Definition 8.17, which means that another element of $U_{\mathcal{K}_G^d}(\mathbf{P}_n^d, \mathsf{KA}(\mathcal{K}_G^d) \setminus \mathbf{N}_n^d)$ has been encountered in the list of goals. Note that **SLG($\mathcal{O}$)** selects literals in some order, while the greatest unfounded set U just refers to some other element in U. Consequently, it may happen that we have to evaluate some literals first whose evaluation is only known in an iteration step m with $m > n$. But this does not cause any problem. Negative literals are simply delayed(DELAYING), while positive literals are processed (NEW SUBGOAL and so on): if a positive literal can eventually be resolved, then it is removed from the list of goals of a child. Otherwise, we obtain an even larger unfounded set. In both cases, once no further operation can be applied, the set U can be completed, and, by Definition 8.19, we derive **not** $L_1^d \in I_{\mathcal{F}}$.

Now, suppose that $L \notin \mathsf{KA}(\mathcal{K}_G^d)$ and $\mathcal{O} \cup \mathbf{P}_n^d \models L$ or $\mathcal{O}^d \cup \mathbf{P}_n^d \models L$. We can can construct a tree starting with $L : - \mid L$ and apply ORACLE RESOLUTION until we get a child $L : - \mid \mathit{Goals}$ such that $\mathit{Goals} \subseteq \mathbf{P}_n^d$, which has to exist. We apply POSITIVE RETURN to all positive literals in Goals, which is possible by the induction hypothesis (1) thus deriving the answer $L : - \mid$, from which we conclude$L \in I_{\mathcal{F}}$.

(Soundness): We show by induction on n that, for $L \in \mathsf{KA}(\mathcal{K}_G^d)$, if L is a positive literal, then $L \in I_{\mathcal{F}_n}$ implies that $L \in \mathbf{P}_\omega^d$, and if $L = \mathbf{not}\, L_1^d$ is a negative literal, then $L \in I_{\mathcal{F}_n}$ implies that $L_1^d \notin \mathbf{N}_\omega^d$, and, for $L \notin \mathsf{KA}(\mathcal{K}_G^d)$, if $L \in I_{\mathcal{F}}$, then $\mathcal{O} \cup \mathbf{P}_\omega^d \models L$ or $\mathcal{O}^d \cup \mathbf{P}_\omega^d \models L$.

The induction base holds trivially, since $\mathcal{F}_0$ is empty. So assume the property holds for n. We show, for $\mathcal{F}_{n+1}$, that the property holds by a case distinction on the operation applied on (cf. Definition 8.21).

1. New Subgoal: This operation creates a new tree and does alone not alter $\mathcal{F}$, i.e., if $L \in \mathcal{F}_{n+1}$, then $L \in \mathcal{F}_n$, and the property holds by the induction hypothesis.

2. Program Clause Resolution: A new child is created for the root $S : - \mid S$. If this child has an empty list of goals, then a rule with empty body was used to create this child. Now, if L is a positive literal with $L = S$, then $L \in \mathcal{F}_{n+1}$. But then, $L \in \mathbf{P}_\omega^d$ since there is a fact L in $\mathcal{K}_G^d$. Alternatively, if the list of children is not empty, then $L \in \mathcal{F}_{n+1}$ implies $L \in \mathcal{F}_n$, and the property holds by the induction hypothesis.

3. Oracle Resolution: A new child is created for the root $S : - \mid S$ by means of the oracle. If the returned list of goals is empty, then the oracle allows us to derive the root directly, and $\mathcal{O} \cup I_{\mathcal{F}_n} \models L$ or $\mathcal{O}^d \cup I_{\mathcal{F}_n} \models L$. In this case, if L is a positive literal with $L = S$, then $L \in \mathcal{F}_{n+1}$. If $L \in \mathsf{KA}(\mathcal{K}_G^d)$, then $L \in \mathbf{P}_\omega^d$, since the operator $D_{\mathcal{K}_G^d}$ together with all $L' \in \mathcal{F}_n$, for which $L' \in \mathbf{P}_\omega^d$ holds by the induction hypothesis, allows us to derive L. If $L \notin \mathsf{KA}(\mathcal{K}_G^d)$, then $\mathcal{O} \cup \mathbf{P}_\omega^d \models L$ or $\mathcal{O}^d \cup \mathbf{P}_\omega^d \models L$ holds since we have that, for all $L' \in \mathcal{F}_n$, $L' \in \mathbf{P}_\omega^d$ holds by the induction hypothesis. Alternatively, if the list of goals is not empty, then $L \in \mathcal{F}_{n+1}$ implies $L \in \mathcal{F}_n$, and the property holds by the induction hypothesis.

4. Positive Return: If the resolved goal is the last remaining, then the outcome of the operation is an unconditional answer. Suppose the answer template is equal to L. We can trace back this child to the immediate child of the root. All goals in this particular child have already been resolved, so that, by the induction hypothesis, all positive literals L appear in $\mathbf{P}_\omega^d$ and all negative literals $\mathbf{not}\, L_1^d$ do not appear in $\mathbf{N}_\omega^d$. But then the property holds, no matter whether $L \in \mathsf{KA}(\mathcal{K}_G^d)$ or not. Alternatively, if the list of goals (including delayed ones) is not empty, then $L \in \mathcal{F}_{n+1}$ implies $L \in \mathcal{F}_n$, and the property holds by the induction hypothesis.

5. Negative Return

(a) Negation Success: The argument is exactly the same as for Positive Return, only now the last goal is a negative literal.

(b) Negation Failure: This operation fails one child. However, it does alone not contribute to $\mathcal{F}$, i.e., if $L \in \mathcal{F}_{n+1}$, then $L \in \mathcal{F}_n$, and the property holds by the induction hypothesis.

6. Delaying: This operation does at best provide a conditional answer. As such it does not affect $\mathcal{F}$ alone. Therefore, if $L \in \mathcal{F}_{n+1}$, then $L \in \mathcal{F}_n$, and the property holds by the induction hypothesis.

7. Simplification:

   (a) The first simplification case corresponds exactly to Negation Failure, only here the failure occurs in the delay list.

   (b) The second simplification case corresponds exactly to Negation Success, only now the success occurs in the delay list.

8. Completion: This operation only affects $\mathcal{F}$ if some A is in the ground instantiation of a completely evaluated literal in $\mathcal{F}$ and A is not in the ground instantiation of any answer in a tree in $\mathcal{F}$. In other words, this operation introduces $\mathbf{not}\, L'$ to $\mathcal{F}$. In particular, consider $L = \mathbf{not}\, L_1^d$ as a negative literal and $L \in I_{\mathcal{F}_{n+1}}$. Thus, the tree for L_1^d does not contain any answer but also no further operation can be applied, i.e., in each child, there is (at least) one literal that can not be resolved or it is failed. This matches the condition of the greatest unfounded set and we obtain that $L_1^d \notin \mathbf{N}_\omega^d$. For all other cases, if $L \in \mathcal{F}_{n+1}$, then $L \in \mathcal{F}_n$, and the property holds by the induction hypothesis.

9. Answer Completion: This operation does not affect $\mathcal{F}$. It only creates failure nodes for unsupported answers, which may be evaluated in a subsequent Completion operation. Consequently, if $L \in \mathcal{F}_{n+1}$, then $L \in \mathcal{F}_n$, and the property holds by the induction hypothesis.

□

Given the soundness of our procedure w.r.t. $\mathbf{P}_\omega^d$ and $\mathbf{N}_\omega^d$, and thus w.r.t. the well-founded partition, from which we obtain the well-founded MKNF model, we can show that soundness carries over to the semantics of MKNF knowledge bases of [Motik and Rosati, 2010].

Corollary 8.27. *Let $\mathcal{K}$ be a consistent hybrid MKNF knowledge base and L be a literal which appears in $\mathcal{K}_G^d$. If $L \in I_\mathcal{F}$ ($L = \mathbf{not}\ L_1^d \in I_\mathcal{F}$ respectively), where $I_\mathcal{F}$ is induced by the forest $\mathcal{F}$ of an* **SLG($\mathcal{O}$)** *evaluation of $\mathcal{K}_G^d$ for query $q = L$, then L (**not** L_1 respectively) is derivable from all two-valued MKNF models of $\mathcal{K}$.*

In addition to the interpretation of the final forest $I_\mathcal{F}$ being sound with respect to the two-valued MKNF models of a given $\mathcal{K}$, the conditional answers in $\mathcal{F}$ can be seen as a well-founded reduct of the rules in $\mathcal{K}$, augmented with conditional answers derived by Oracle Resolution operations. As a result, the final forest can be seen as a *residual program*: a sound transformation not only of the rules, but of information from the oracle, and can be used to construct a partial two-valued MKNF model.[1]

Regarding complexity, it is clear that the complexity of the whole procedure **SLG($\mathcal{O}$)** depends on the complexity of the oracle, and also on the number of results returned by each call to the oracle. Clearly, the complexity associated to the computation of one result of the oracle function is a lower-bound of the complexity of **SLG($\mathcal{O}$)**. Moreover, even if the computation of one result of the oracle is tractable, the (data) complexity of **SLG($\mathcal{O}$)** may still be exponential if exponentially many solutions are generated by the oracle, e.g., if returning all supersets of a solution. This is so, because our definition of the oracle is quite general, and in order to prove interesting complexity results some assumptions must be made about the oracle. We start by defining a correct partial oracle:

Definition 8.28. *Let $\mathcal{K}_G^d = (\mathcal{O}, \mathcal{O}^d, \mathcal{P}_G^d)$ be a doubled, hybrid MKNF knowledge base, S a goal, and L a set of ground atoms which appear in at least one rule head in $\mathcal{P}_G$ (called program atoms). A* partial oracle *for $\mathcal{K}_G^d$, denoted $pT_\mathcal{O}$, is a relation $pT_\mathcal{O}(I_{\mathcal{F}_n}, S, L)$ such that if $pT_\mathcal{O}(I_{\mathcal{F}_n}, S, L)$, then*

$$\mathcal{O} \cup I_{\mathcal{F}_n}^+ \cup L \models S \text{ and } \mathcal{O} \cup I_{\mathcal{F}_n}^+ \cup L \text{ consistent; or}$$
$$\mathcal{O}^d \cup I_{\mathcal{F}_n}^+ \cup L \models S \text{ and } \mathcal{O}^d \cup I_{\mathcal{F}_n}^+ \cup L \text{ consistent.}$$

A partial oracle $pT_\mathcal{O}$ is correct *w.r.t. $compT_\mathcal{O}$ iff, for all MKNF-consistent $\mathcal{K}_G^d$, replacing $compT_\mathcal{O}$ in* **SLG($\mathcal{O}$)** *with $pT_\mathcal{O}$ succeeds for exactly the same set of queries.*

[1] [Chen and Warren, 1996] discusses these transformational aspects of SLG resolution, which are preserved in **SLG($\mathcal{O}$)**, while the XSB manual (`http://xsb.sourceforge.net/`) discusses how the residual program can serve as input to an ASP solver.

Note that the complete oracle is indeed generating unnecessarily many answers, and it can be replaced by a partial one that assures correctness. E.g., consider a partial oracle that does not return supersets of other results. Such a partial oracle is obviously correct. A further improvement on efficiency is the restriction to consistent sets $\mathcal{O} \cup I^+_{\mathcal{F}_n} \cup L$ and $\mathcal{O}^d \cup I^+_{\mathcal{F}_n} \cup L$. If the knowledge base is MKNF-consistent, then looking for derivations based on inconsistencies is pointless anyway: we would just create a potentially large number of children none of which would result in an unconditional answer. In this sense, partial oracles are limited to meaningful derivations. In the case of an MKNF-inconsistent knowledge base, things get a bit more complicated.

Example 8.29. *Consider again the already doubled knowledge base from Example 8.12.*

$$\begin{array}{ll} \mathtt{Q} \sqsubseteq \neg \mathtt{R} & \mathtt{Q}^{\mathtt{d}} \sqsubseteq \neg \mathtt{R}^{\mathtt{d}} \\ \mathtt{p}(\mathtt{a}) \leftarrow \mathbf{not}\, \mathtt{p}^{\mathtt{d}}(\mathtt{a}) & \mathtt{p}^{\mathtt{d}}(\mathtt{a}) \leftarrow \mathbf{not}\, \mathtt{p}(\mathtt{a}) \\ \mathtt{Q}(\mathtt{a}) \leftarrow & \mathtt{Q}^{\mathtt{d}}(\mathtt{a}) \leftarrow \mathbf{not}\, \mathtt{NQ}(\mathtt{a}) \\ \mathtt{R}(\mathtt{a}) \leftarrow \mathbf{not}\, \mathtt{R}^{\mathtt{d}}(\mathtt{a}) & \mathtt{R}^{\mathtt{d}}(\mathtt{a}) \leftarrow \mathbf{not}\, \mathtt{R}(\mathtt{a}), \mathbf{not}\, \mathtt{NR}(\mathtt{a}) \end{array}$$

Cf. the computation in Example 8.12, $\mathtt{p}(\mathtt{a})$, $\mathtt{Q}(\mathtt{a})$, *and* $\mathtt{R}(\mathtt{a})$ *are true in the sequence* $\mathbf{P}^d_\omega$ *while* $\mathtt{p}^{\mathtt{d}}(\mathtt{a})$, $\mathtt{Q}^{\mathtt{d}}(\mathtt{a})$, *and* $\mathtt{R}^{\mathtt{d}}(\mathtt{a})$ *are false in the sequence* $\mathbf{N}^d_\omega$. *The same results are derivable with a complete oracle.* $\mathtt{Q}(\mathtt{a})$ *is derivable from the corresponding fact. From that* $\neg\mathtt{R}(\mathtt{a})$ *is derivable and therefore* **not** $\mathtt{R}^{\mathtt{d}}(\mathtt{a})$ *as well. This allows us to obtain* $\mathtt{R}(\mathtt{a})$. *Now,* $\mathtt{Q}(\mathtt{a})$ *and* $\mathtt{R}(\mathtt{a})$ *together with* $\mathcal{O}$ *are inconsistent from which we can derive* $\mathtt{p}(\mathtt{a})$, *but also* $\neg\mathtt{Q}(\mathtt{a})$ *and* $\neg\mathtt{p}(\mathtt{a})$. *Consequently,* **not** $\mathtt{p}^{\mathtt{d}}(\mathtt{a})$ *and* **not** $\mathtt{Q}^{\mathtt{d}}(\mathtt{a})$ *hold as well, i.e., everything is supposedly true and false at the same time.*

If we limit ourselves to the partial (consistent) oracle, then we no longer derive $\mathtt{p}(\mathtt{a})$, **not** $\mathtt{Q}(\mathtt{a})$, *or* **not** $\mathtt{p}(\mathtt{a})$. *In this case,* $\mathtt{R}(\mathtt{a})$ *is still true and false (inconsistent), but* $\mathtt{Q}(\mathtt{a})$ *is true, and* $\mathtt{p}(\mathtt{a})$ *is undefined.*

Thus, the usage of such a partial oracle hides more any occurring MKNF-inconsistency and demonstrates a somewhat paraconsistent behavior instead.

This example also shows why correctness of a partial oracle is only checked w.r.t. MKNF-consistent knowledge bases. For MKNF-inconsistent knowledge bases the derivation relation is not identical in general.

By making assumptions on the complexity and number of results of an oracle, complexity results of **SLG($\mathcal{O}$)** are obtained.

Theorem 8.30. *Let $pT_{\mathcal{O}}$ be a correct partial oracle for the hybrid MKNF knowledge base $\mathcal{K}_G^d$, such that for every goal S, the cardinality of $pT_{\mathcal{O}}(I_{\mathcal{F}_n}, S, L)$ is bound by a polynomial on the number of program atoms. Moreover, assume that computing each element of $pT_{\mathcal{O}}$ is decidable with data complexity $\mathcal{C}$. Then, the* **SLG($\mathcal{O}$)** *evaluation of a query in $\mathcal{K}_G^d$ is decidable with data complexity PTIME$^{\mathcal{C}}$.*

Proof. Decidability is guaranteed by Theorem 8.24. As for complexity, first note that, given the polynomial data complexity of SLG [Chen and Warren, 1996], only a polynomial number of calls is made to the oracle. Moreover, since the cardinality of $pT_{\mathcal{O}}(I_{\mathcal{F}_n}, S, L)$ is bound by a polynomial, and each of the calls to the oracle can be seen as adding a new program rule (the result of ORACLE RESOLUTION operation), only polynomially many such rules are added. Now, computing each such rule amounts to a call to the oracle, which by hypothesis is decidable and with data complexity $\mathcal{C}$. So, the overall data complexity is PTIME$^{\mathcal{C}}$. □

In particular, this means that if the partial oracle is tractable, and only with polynomial many results, then **SLG($\mathcal{O}$)** is also tractable.

We have thus defined a general procedure to query hybrid MKNF knowledge bases. This procedure builds on SLG resolution with tabling for rules and extends it with an oracle to any arbitrary decidable DL. Under certain conditions, the computational complexity carries over from the bottom-up computation. These conditions can be achieved with an appropriately chosen partial oracle. Such oracles are defined in the next chapter, thus arguing in favor of the efficiency of our presented querying mechanism.

9

Querying Tractable Hybrid MKNF Knowledge Bases

Clearly, for a hybrid MKNF knowledge base whose ontology part is a tractable fragment, it is possible to come up with a correct partial oracle. Basically, all that needs to be done is to proceed with the usual entailment method, assuming that all program atoms hold, and collecting them for the oracle result. To guarantee that the number of solutions of the oracle is bound by a polynomial, and still maintaining correctness, is a bit more difficult. It amounts to finding a procedure that returns less results, and at the same time does not damage the completeness proof (similar to that of Theorem 8.26). At least for the tractable case this is possible, albeit the oracle being the (polynomial complexity) bottom-up procedure that calculates the well-founded partition. This approach is, however, somewhat counterproductive to the whole idea of a top-down querying mechanism: we could simply use the bottom-up procedure in the first place to compute the model and store the results in a database which we then query on demand. In this chapter, we show that these conditions are indeed realistic. We provide concrete procedures, with practical usage, by defining oracles for $\mathcal{REL}$, DLP and $DL\text{-}Lite_{\mathcal{R}}$, i.e., the three tractable DL fragments, which we presented in Section 3.3, that are underlying the OWL 2 profiles that are part of the W3C recommendations [Hitzler et al., 2009a] for the Semantic Web. We show that the oracles thus defined are correct w.r.t. the general procedure and maintain the polynomial data complexity.

We follow the structure in Section 3.3 and present at first an oracle for $\mathcal{REL}$ (Section 9.1). We continue with an oracle for $DL\text{-}Lite_{\mathcal{R}}$ (Section 9.2), and finish the chapter

with an oracle for DLP (Section 9.3).

9.1 An Oracle for $\mathcal{REL}$

When defining an oracle for $\mathcal{REL}$ we could simply try to use the algorithm for subsumption presented in [Baader et al., 2005]: reduce instance checking to subsumption and return the desired set of atoms which, when proven, would ensure the derivability of the initial query. However, apart from the technical problems we would have to face, such as how to obtain these sets of atoms whose truth allows us to prove the initial query, this would mean that we would have to run the entire subsumption algorithm for each query posed to the oracle in $\mathcal{REL}$.

We therefore proceed differently. We still use the algorithm for subsumption from [Baader et al., 2005] to compute the complete class hierarchy of the $\mathcal{REL}$ knowledge base, but we use it only once, as a preprocessing step for the derivable information of the ontology. Then we take the obtained results together with the $\mathcal{REL}$ KB and simplify them by removing all statements that are redundant when looking for instances of classes in a top-down manner. The result, including the ABox, is then turned into a set of rules which can be used in a top-down manner, using SLG alone to yield the desired oracle. Moreover, this way we can straightforwardly combine these transformed rules with the ones in the knowledge base and, using $\mathbf{SLG}(\mathcal{O})$ as defined in Chapter 8, obtain a single top-down procedure querying a doubled MKNF knowledge base where the ontology is described in $\mathcal{REL}$.

To achieve an appropriate oracle, we first recall the (simplified) subsumption algorithm for $\mathcal{REL}$. Then, using this algorithm, we devise a simplification of the $\mathcal{REL}$ knowledge base, which is translated into rules.

9.1.1 Deciding Subsumption in $\mathcal{REL}$

The main inference problem considered in [Baader et al., 2005] for $\mathcal{SROEL}$ (also called $\mathcal{EL}^{++}$) is concept subsumption, and, here, we recall the algorithm presented in [Baader et al., 2005], restricted to $\mathcal{REL}$. For that, we start by recalling a normal form for $\mathcal{REL}$ knowledge bases.

Definition 9.1. *An $\mathcal{REL}$ knowledge base is in* normal form *if*

1. *all GCIs have one of the following forms, where* C_1, $C_2 \in \mathsf{N_C} \cup \{\top\}$ *and* $D \in \mathsf{N_C} \cup \{\bot, \top\}$*:*

$$\begin{array}{llll} (1) & C_1 \sqsubseteq D & (3) & \exists R.C_1 \sqsubseteq D \\ (2) & C_1 \sqcap C_2 \sqsubseteq D & (4) & C_1 \sqsubseteq \exists R.C_2 \end{array}$$

2. *all RIAs are of the form* $R \sqsubseteq S$ *or* $R_1 \circ R_2 \sqsubseteq S$*.*

By appropriately introducing new concept and role names, any $\mathcal{REL}$ knowledge base can be turned into normal form and, as shown in [Baader et al., 2005], this transformation can be done in linear time. So, from now on, we assume that any $\mathcal{REL}$ knowledge base is in normal form.

The subsumption algorithm for $\mathcal{REL}$ ([Baader et al., 2005]) applies a set of completion rules to compute the entire class hierarchy, i.e., all subsumption relationships between all pairs of concept names occurring in the knowledge base. In detail, given a normalized $\mathcal{REL}$ knowledge base, the algorithm computes:

- a mapping S from $\mathsf{N_C} \cup \{\top\}$ to a subset of $\mathsf{N_C} \cup \{\bot, \top\}$; and
- a mapping T from $\mathsf{N_R}$ to a binary relation on $\mathsf{N_C} \cup \{\top\}$.

These mappings materialize implicit relations as follows:

(I1) $D \in S(C)$ implies that $C \sqsubseteq D$,

(I2) $(C, D) \in T(R)$ implies that $C \sqsubseteq \exists R.D$.

These mappings are appropriately initialized:

- $S(C) := \{C, \top\}$ for each $C \in \mathsf{N_C} \cup \{\top\}$
- $T(R) := \emptyset$ for each $R \in \mathsf{N_R}$

Then the completion rules are applied to extend $S(C)$ and $T(R)$ until no more rule applies.

CR1 If $C' \in S(C)$, $C' \sqsubseteq D \in \mathcal{C}$, and $D \notin S(C)$
then $S(C) := S(C) \cup \{D\}$

CR2 If $C_1, C_2 \in S(C)$, $C_1 \sqcap C_2 \sqsubseteq D \in \mathcal{C}$, and $D \notin S(C)$
then $S(C) := S(C) \cup \{D\}$

CR3 If $C' \in S(C)$, $C' \sqsubseteq \exists R.D \in \mathcal{C}$, and $(C, D) \notin T(R)$
then $T(R) := T(R) \cup \{(C, D)\}$

CR4 If $(C, D) \in T(R)$, $D' \in S(D)$, $\exists R.D' \sqsubseteq E \in \mathcal{C}$, and $E \notin S(C)$
then $S(C) := S(C) \cup \{E\}$

CR5 If $(C, D) \in T(R)$ $\bot \in S(D)$, and $\bot \notin S(C)$
then $S(C) := S(C) \cup \{\bot\}$

CR6 If $(C, D) \in T(R)$, $R \sqsubseteq S \in \mathcal{C}$, and $(C, D) \notin T(S)$
then $T(S) := T(S) \cup \{(C, D)\}$

CR7 If $(C, D) \in T(R_1)$, $(D, E) \in T(R_2)$, $R_1 \circ R_2 \sqsubseteq R_3 \in \mathcal{C}$, and $(C, E) \notin T(R_3)$
then $T(R_3) := T(R_3) \cup \{(C, E)\}$

Compared to the completion rules in [Baader et al., 2005], we omitted the four completion rules related to nominals and concrete domains, since $\mathcal{REL}$ does not include these constructors.

It was shown in [Baader et al., 2005] that this algorithm terminates in polynomial time and that it is correct.

Lemma 9.2. *Let S be the mapping obtained after the subsumption algorithm for the normalized $\mathcal{REL}$ knowledge base KB has terminated, and A and B concept names occurring in KB. Then $A \sqsubseteq B$ iff the following condition holds:*

- $S(A) \cap \{B, \bot\} \neq \emptyset$.

This means that either A is an unsatisfiable concept or the subsumption relationship is explicitly present in the mapping S. Note that the restriction to $\mathcal{REL}$ simplifies the condition in Lemma 9.2 of [Baader et al., 2005].[1]

9.1.2 Simplifying the Ontology

Given a normalized $\mathcal{REL}$ KB $\mathcal{O}$ (Definition 9.1), the first transformation step is to apply the subsumption algorithm to $\mathcal{O}$ and obtain the mappings S and T computed by it. In particular, we obtain via S all the subsumption relationships implicitly or explicitly present in $\mathcal{O}$. In fact, it is easy to see that the initialization of $C \in S(C)$

[1] There, another condition related to nominals appears.

for each $C \in \mathsf{N_C} \cup \{\top\}$ ensures that each GCI of the form (1) of the normal form of Definition 9.1 ($C_1 \sqsubseteq D$) is also obtained by $D \in S(C_1)$, and each GCI of the form (4) ($C_1 \sqsubseteq \exists R.C_2$) is obtained by $(C_1, C_2) \in T(R)$.[1]

It follows immediately, that we can ignore all GCIs of the form (1) and (4) as long as we have the complete mappings S and T of the subsumption algorithm available. But we can simplify even more.

Example 9.3. *Consider the hybrid MKNF knowledge base with $\mathcal{O}$ in $\mathcal{REL}$, containing one rule, and some facts.*

$$\begin{aligned} \mathtt{C} &\sqsubseteq \exists \mathtt{R}.\mathtt{D} & \mathtt{G}(\mathtt{x}) &\leftarrow \mathtt{D}(\mathtt{x}) \\ \exists \mathtt{R}.\mathtt{C} &\sqsubseteq \mathtt{D} & \mathtt{C}(\mathtt{a}). \quad & \mathtt{C}(\mathtt{b}). \\ \mathtt{C}_1 \sqcap \mathtt{C}_2 &\sqsubseteq \mathtt{D} & \mathtt{R}(\mathtt{a},\mathtt{b}). & \end{aligned}$$

Now consider that we want to know whether $\mathtt{G}(\mathtt{a})$ *holds. There is only one rule that allows us to derive* $\mathtt{G}(\mathtt{a})$*, and this requires that* $\mathtt{D}(\mathtt{a})$ *is derivable. Obviously, if we have* $\mathtt{C}_1(\mathtt{a})$ *and* $\mathtt{C}_2(\mathtt{a})$ *then* $\mathtt{D}(\mathtt{a})$ *holds as well. But this information is currently not present in the knowledge base. If we check the second GCI then obtaining* $\mathtt{D}(\mathtt{a})$ *requires finding* $\mathtt{R}(\mathtt{a},\mathtt{x})$ *and* $\mathtt{C}(\mathtt{x})$ *which appear as facts in the rule part, for* $\mathtt{x} = \mathtt{b}$*. Intuitively, we want the oracle to transform the query* $\mathtt{D}(\mathtt{a})$ *into an* **SLG($\mathcal{O}$)** *node* $\mathtt{D}(\mathtt{a}) : - \mid \mathtt{R}(\mathtt{a},\mathtt{x}), \mathtt{C}(\mathtt{x})$*, the goals of which can then be resolved, leading to a derivation of* $\mathtt{D}(\mathtt{a})$*.*

Next, suppose we alternatively query for $\mathtt{G}(\mathtt{b})$*, and subsequently query the oracle for* $\mathtt{D}(\mathtt{b})$*. Then the second GCI does not allow us to derive* $\mathtt{D}(\mathtt{b})$ *because there is no* $\mathtt{R}(\mathtt{b},\mathtt{x})$ *for some* $\mathtt{x}$ *derivable; the third does not allow us to derive* $\mathtt{D}(\mathtt{b})$ *because there are no individuals known to hold in* $\mathtt{C}_1$ *or* $\mathtt{C}_2$*. But even using the first GCI does not allow us to derive* $\mathtt{D}(\mathtt{b})$*: while* $\mathtt{C}(\mathtt{a})$ *holds and we know that there is an explicit relation* $\mathtt{R}(\mathtt{a},\mathtt{b})$ *in the knowledge base, the semantics of $\mathcal{O}$ (and descriptive first-order semantics in general) does not allow to derive* $\mathtt{D}(\mathtt{b})$*, since* $\mathtt{D}(\mathtt{b})$ *does not hold in all models of $\mathcal{O}$ - there are models where* $\mathtt{R}(\mathtt{a},\mathtt{i})$ *and* $\mathtt{D}(\mathtt{i})$ *hold for some individual* $\mathtt{i}$ *not appearing in the knowledge base.*

Clearly in a normalized $\mathcal{O}$, GCIs of the form (3) ($\exists R.C_1 \sqsubseteq D$) and (2) ($C_1 \sqcap C_2 \sqsubseteq D$) – and therefore also of the form (1) – can be used to derive information when answering an (instance) query. On the other hand, the example implies that GCIs of the form (4) ($C_1 \sqsubseteq \exists R.C_2$) do not contribute to drawing this kind of conclusions. We now formalize this observation.

[1] Cf. the completion rules $CR1$ and $CR3$, which precisely add each such explicit GCIs to the appropriate mapping.

For simplicity of notation, we start by transforming all the mappings obtained from the algorithm into GCIs, and then we remove all GCIs of the form (4).

Definition 9.4. *Let $\mathcal{O}$ be in $\mathcal{REL}$ and S and T the mappings obtained from the subsumption algorithm. We obtain the* completed $\mathcal{REL}$ KB $\mathcal{O}'$ *from $\mathcal{O}$ by adding for each $D \in S(C)$ a GCI $C \sqsubseteq D$ to $\mathcal{O}'$ and for each $(C, D) \in T(R)$ a GCI $C \sqsubseteq \exists R.D$ to $\mathcal{O}'$.*

Let $\mathcal{O}$ be a completed $\mathcal{REL}$ KB. We define the reduced $\mathcal{REL}$ KB $\mathcal{O}'$*, which is obtained from the completed $\mathcal{REL}$ KB $\mathcal{O}$ by removing all GCIs of form (4).*

It is straightforward to see that the transformation from $\mathcal{O}$ to the completed $\mathcal{REL}$ KB $\mathcal{O}'$ simply allows us to disregard the mappings S and R obtained by the algorithm of subsumption without losing any of the subset relationships contained in these mappings.

Now we have to show that a reduced $\mathcal{REL}$ KB $\mathcal{O}$, which in general does not preserve the semantics of $\mathcal{O}$, is still suitable for answering queries of the form $C(a)$ or $R(a, b)$.

Proposition 9.5. *Let $\mathcal{O}$ be a completed $\mathcal{REL}$ KB, and $\mathcal{O}'$ the reduced $\mathcal{REL}$ KB obtained from $\mathcal{O}$. Then the two conditions hold.*

(i) a is an instance of concept C in $\mathcal{O}$ iff a is an instance of concept C in $\mathcal{O}'$.

(ii) (a, b) is an instance of role R in $\mathcal{O}$ iff (a, b) is an instance of role R in $\mathcal{O}'$.

Proof. For (i) we have to show that $a^{\mathcal{I}} \in C^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{O}$ iff $a^{\mathcal{I}'} \in C^{\mathcal{I}'}$ for every model $\mathcal{I}'$ of $\mathcal{O}'$; for (ii) we have to show that $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{O}$ iff $(a^{\mathcal{I}'}, b^{\mathcal{I}'}) \in R^{\mathcal{I}'}$ for every model $\mathcal{I}'$ of $\mathcal{O}'$. We are going to show the argument for (i); the case of (ii) follows analogously.
$'\Leftarrow'$: follows directly from monotonicity: adding GCIs of the form (4) does not invalidate any drawn conclusions, i.e., if $a^{\mathcal{I}'} \in C^{\mathcal{I}'}$ for every model $\mathcal{I}'$ of $\mathcal{O}'$ then adding GCIs of the form (4) can only reduce the models of $\mathcal{O}$ and never increase. We conclude $a^{\mathcal{I}} \in C^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{O}$.
$'\Rightarrow'$: Suppose that $a^{\mathcal{I}} \in C^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{O}$. If none of the GCIs of the form (4) contains the concept name C then we can remove them all and $a^{\mathcal{I}'} \in C^{\mathcal{I}'}$ for every model $\mathcal{I}'$ of $\mathcal{C}'$. The same argument applies if C appears only on the left hand side of such GCIs. So assume C appears on the right hand side of at least one such GCI $C_1 \sqsubseteq \exists R.C$. However, even if there is an individual i such that $i^{\mathcal{I}} \in C_1^{\mathcal{I}}$ and $(i^{\mathcal{I}}, a^{\mathcal{I}}) \in R^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{O}$, then the removal of GCIs of the form (4) is not relevant for that since $C_1 \sqsubseteq \exists R.C$ does not allow us to derive that the unknown individual is a fixed one. We can thus conclude that $a^{\mathcal{I}'} \in C^{\mathcal{I}'}$ for every common model $\mathcal{I}'$ of $\mathcal{O}'$. □

Having proven that completion and reduction of $\mathcal{REL}$ KBs does not alter the derivability of instance queries, we can take a short cut: instead of completing the $\mathcal{REL}$ KB $\mathcal{O}$ we can directly remove all GCIs of the form (4) and discard the mapping R. We then complete the $\mathcal{REL}$ KB $\mathcal{O}$ only w.r.t. the mapping S and obtain the reduced $\mathcal{REL}$ KB $\mathcal{O}'$.

Corollary 9.6. *Let $\mathcal{O}$ be a $\mathcal{REL}$ KB and S and T the mappings obtained from the subsumption algorithm. We obtain the reduced $\mathcal{REL}$ KB $\mathcal{O}'$ from $\mathcal{O}$ by removing all GCIs of the form (4) from $\mathcal{O}$ and by adding for each $D \in S(C)$ a GCI $C \sqsubseteq D$.*

9.1.3 Transformation into Rules

Now, we show how to transform the reduced $\mathcal{REL}$ KB into rules in such a way that running the SLG procedure on the obtained set of rules yields an oracle that can be used in $\mathbf{SLG}(\mathcal{O})$. Special care must be taken with inconsistencies and with the fact that if an atom is proven false in the ontology, then its negation also holds in the rules. Note that this is achieved in Chapter 8 by querying for classically negated atoms, but these are outside the syntax of $\mathcal{REL}$ even though a restricted form of negation is achievable via $\bot$.

Regarding inconsistencies, there are two kinds which can appear in the three-valued hybrid MKNF semantics: either the ontology alone is inconsistent, or there is an inconsistency resulting from the interaction of the rules and the ontology. In the first case, there is not much to be done. An inconsistent ontology has no models and we can simply derive anything from it. We therefore admit an a-priori consistency check of the ontology alone, and proceed only if it succeeds, i.e., we limit ourselves to a consistent ontology.[1] For the second case, the bottom-up computation allows us to detect such problems, but in $\mathbf{SLG}(\mathcal{O})$ we are limited to finding atoms that are true and false at the same time, i.e., if for some $\mathtt{C}(\mathtt{a})$ both queries $\mathtt{C}(\mathtt{a})$ and $\mathbf{not}\,\mathtt{C}^{\mathtt{d}}(\mathtt{a})$[2] are answered with 'yes', then the combined KB is inconsistent. This can, of course, not be complete for a partial oracle as shown in Example 8.29, so that we obtain a paraconsistent behavior. To carry over this behavior to a transformation into rules, we have to take into con-

[1] Note that ontologies in $\mathcal{REL}$ can in fact be inconsistent: consider a GCI $\mathtt{C} \sqsubseteq \bot$ and an assertion $\mathtt{C}(\mathtt{a})$ in the ABox.

[2] Recall that we use the doubled predicate for determining falsity.

sideration the transformation presented in Definition 8.3 and their effect on the $\mathcal{REL}$ KB.

Regarding classical negation, we solve the problem in a specific way. In $\mathbf{SLG}(\mathcal{O})$, the special negative literals **not** $NH(t_i)$ are used to call $\neg H(t_i)$. Since this is not expressible in $\mathcal{REL}$ we simply consider **not** $NH(t_i)$ as normal negative literals, and transform $\mathcal{O}$ into rules such that **not** $NH(t_i)$ holds if $\neg H(t_i)$ holds. More precisely, if $H \sqsubseteq \bot$, then $NH(t_i)$ holds.

We are now ready to define the transformation of the reduced $\mathcal{REL}$ KB $\mathcal{O}$ into a set of already doubled rules (see Definition 8.1).

Definition 9.7. *Let $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ be a hybrid MKNF knowledge base with a consistent, reduced $\mathcal{REL}$ KB $\mathcal{O}$. We define $\mathcal{P}^d_{\mathcal{O}}$ from $\mathcal{O}$, where C,D, C_1 and C_2 are concept names, R, S, T are role names, and a, b are individual names, as the smallest set containing:*

(a1) *for each $C(a) \in \mathcal{A}$: $C(a) \leftarrow$ and $C^d(a) \leftarrow$ **not** $NC(a)$.*

(a2) *for each $R(a,b) \in \mathcal{A}$: $R(a,b) \leftarrow$ and $R^d(a,b) \leftarrow$ **not** $NR(a,b)$.*

(c1) *for each GCI $C \sqsubseteq D$: $D(x) \leftarrow C(x)$ and*
$D^d(x) \leftarrow C^d(x), \mathbf{not}\, ND(x)$.

(c2) *for each $C_1 \sqcap C_2 \sqsubseteq D$: $D(x) \leftarrow C_1(x), C_2(x)$ and*
$D^d(x) \leftarrow C_1^d(x), C_2^d(x), \mathbf{not}\, ND(x)$.

(c3) *for each $\exists R.C \sqsubseteq D$: $D(x) \leftarrow R(x,y), C(y)$ and*
$D^d(x) \leftarrow R^d(x,y), C^d(y), \mathbf{not}\, ND(x)$.

(r1) *for each RIA $R \sqsubseteq S$: $S(x,y) \leftarrow R(x,y)$ and*
$S^d(x,y) \leftarrow R^d(x,y), \mathbf{not}\, NS(x,y)$.

(r2) *for each RIA $R \circ S \sqsubseteq T$: $T(x,z) \leftarrow R(z,y), S(y,z)$ and*
$T^d(x,z) \leftarrow R^d(x,y), S^d(y,z), \mathbf{not}\, NT(x,z)$.

(i1) *for each $C \sqsubseteq \bot$: $NC(x) \leftarrow$.*

(i2) *for each $C_1 \sqcap C_2 \sqsubseteq \bot$: $NC_2(x) \leftarrow C_1(x)$ and $NC_1(x) \leftarrow C_2(x)$.*

(i3) *for each $\exists R.C \sqsubseteq \bot$: $NC(y) \leftarrow R(x,y)$ and $NR(x,y) \leftarrow C(y)$.*

Note that the cases (i1) to (i3) are used to introduce truth of some $NH(t_i)$. Furthermore, these three cases only produce one rule, since atoms based on predicates of the forms NC^d or NR^d are not required anywhere.

Program $\mathcal{P}^d_{\mathcal{O}}$ can be used as the basis for obtaining a correct partial oracle for $\mathcal{REL}$ knowledge bases, to be integrated in the general procedure of **SLG($\mathcal{O}$)**. Recall that an oracle receives a query S and the already derived (positive) information $I^+_{\mathcal{F}_n}$, and returns a set of atoms L, which if proven, ensure that S is derivable. The general idea of such an oracle for $\mathcal{REL}$ would be to use SLG to query S in the program consisting of $\mathcal{P}^d_{\mathcal{O}}$ and facts for all the atoms in $I^+_{\mathcal{F}_n}$ in such a way that any time an atom also defined in the rules is queried, the atom can succeed, i.e., is removed from the resolvent, and is collected in a set associated to the respective derivation branch.[1] Upon success, the so modified SLG procedure would return the set of collected atoms. The partial oracle would be defined by the relation with the query, the running forest, and the returned set. However, since both the rule part and the oracle itself would be evaluated by an SLG procedure, they can be combined: instead of collecting the atoms in the set, and then calling them in **SLG($\mathcal{O}$)** after the oracle returns a result, one can immediately call the otherwise collected atoms, i.e., the atoms defined in the program. This way, correctness of the so defined partial oracle is equivalent to the correctness of the above transformation. We prove this for MKNF-consistent knowledge bases:

Theorem 9.8. *Let $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ be an MKNF-consistent hybrid MKNF knowledge base with $\mathcal{O}$ in $\mathcal{REL}$. Then $\mathcal{K}_{\mathcal{REL}} = (\emptyset, (\mathcal{P}^d \cup \mathcal{P}^d_{\mathcal{O}}))$ is semantically equivalent to $\mathcal{K}^d = (\mathcal{O}, \mathcal{O}^d, \mathcal{P}^d)$.*

Proof. We have to show that $\mathcal{P}^d_{\mathcal{O}}$ is equivalent to $\mathcal{O}$ and $\mathcal{O}^d$.

The transformations on ABox assertions, (a1) and (a2), on GCIs in $\mathcal{C}$, (c1), (c2), and (c3), and on role inclusions, (r1) and (r2), are semantically equivalent and can be found, e.g., in [Grosof et al., 2003]. Since $\mathcal{O}$ contains the original GCIs and $\mathcal{O}^d$ the doubled ones with new predicate names, we also create two rules, one for each of the two DL knowledge bases in $\mathcal{K}^d$. Note that the addition of predicates, such as $NC(x)$, to the body of a rule with head $C^d(x)$ is just done to enforce that whenever $NC(x)$ holds, i.e., $\neg C(x)$, then $C^d(x)$ cannot become true, which is used in the consistent case to enforce coherence. We only have to consider the transformations (i1) to (i3).

[1] An alternative way of viewing this, would be to add to $\mathcal{P}^d_{\mathcal{O}}$ facts for all the atoms defined in the rules, run SLG as usual, but collecting all those facts that were used in the derivation.

(i1) $C \sqsubseteq \bot$: C is unsatisfiable, i.e., $\neg C(x)$ for all x; $\mathcal{O}$ contains a statement that allows us to infer $\neg C(x)$ which corresponds exactly to the fact $NC(x) \leftarrow$.

(i2) $C_1 \sqcap C_2 \sqsubseteq \bot$: the statement expresses disjointness of C_1 and C_2, i.e., $\neg(C_1(x) \wedge C_2(x))$ for all x which is equivalent to $C_1(x) \rightarrow \neg C_2(x)$ and $C_2(x) \rightarrow \neg C_1(x)$; using the correspondences $\neg C_1(x) \rightarrow NC_1(x)$ and $\neg C_2(x) \rightarrow NC_2(x)$.

(i3) $\exists R.C \sqsubseteq \bot$ follows the same argument as (i2). □

For the MKNF-inconsistent case, we should point out that one result of the transformation into rules is that we obtain a somewhat paraconsistent approach: while an inconsistent ontology allows us to derive anything from it, the process of doubling the rules enables us to derive those consequences which do not depend on inconsistent information contained in the KB as presented in Example 8.29. We leave the details of that to future studies.

Finally, we have to show that the process of translating the ontology into rules and reasoning over the combined set of rules with **SLG($\mathcal{O}$)** also preserves the polynomial data complexity.

Theorem 9.9. *Let $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ be a hybrid MKNF knowledge base with $\mathcal{O}$ in $\mathcal{REL}$. An* **SLG($\mathcal{O}$)** *evaluation of a query in $\mathcal{K}_{\mathcal{REL}} = (\emptyset, (\mathcal{P}^d \cup \mathcal{P}^d_{\mathcal{O}}))$ is decidable with data complexity PTime.*

Proof. In Theorem 8.30, the general complexity result is given for a correct partial oracle. Since evaluation is done with respect to a combined set of rules, i.e., we integrate the rules, the whole process of querying the oracle and returning a limited amount of answers is ensured by the evaluation of rules with tabling which is known to have a data complexity PTime. Together with the polynomial subsumption algorithm for $\mathcal{REL}$ and the linear transformations to obtain $\mathcal{K}_{\mathcal{REL}}$ we prove the claim. □

9.2 An Oracle for *DL-Lite*$_{\mathcal{R}}$

For defining an oracle for *DL-Lite*$_{\mathcal{R}}$ we rely on the reasoning algorithms provided for *DL-Lite*$_{\mathcal{R}}$ in [Calvanese et al., 2007]. The basic reasoning service to which all others are reduced is satisfiability. Satisfiability of a *DL-Lite*$_{\mathcal{R}}$ KB is checked by evaluating a suitable Boolean first-order logic query w.r.t. a canonical model of its ABox. We recall the construction and evaluation of such a formula, and we note that, for simplicity,

we limit ourselves to basic constructs only, i.e., all constructors that can be considered syntactic sugar in Section 3.3.2, are not mentioned explicitly. This means, that we limit ourselves to concept names, and $\exists R.\top$ on both sides of a GCI, which we simplify to $\exists R$, role names and inverse roles names in RIA and GCI, $\neg$ on the right hand side of GCI and RIA, and, in the ABox, to simple assertions for concept and role names only.

9.2.1 Satisfiability in *DL-Lite*$_\mathcal{R}$

First, we recall the definition of a canonical interpretation of the ABox from [Calvanese et al., 2007].

Definition 9.10. *Let $\mathcal{A}$ be a DL-Lite$_\mathcal{R}$ ABox. We denote by $db(\mathcal{A}) = (\Delta^{db(\mathcal{A})}, \cdot^{db(\mathcal{A})})$ the interpretation defined as follows:*

- $\Delta^{db(\mathcal{A})}$ *is the nonempty set consisting of all constants occurring in $\mathcal{A}$;*
- $a^{db(\mathcal{A})} = a$ *for each constant a;*
- $A^{db(\mathcal{A})} = \{a \mid A(a) \in \mathcal{A}\}$ *for each atomic concept A; and*
- $R^{db(\mathcal{A})} = \{(a, b) \mid R(a, b) \in \mathcal{A}\}$ *for each atomic role R.*

It is easy to see that $db(\mathcal{A})$ is in fact a model of $\mathcal{A}$ and a minimal one [Calvanese et al., 2007].

This forms the basis for checking satisfiability in *DL-Lite*$_\mathcal{R}$ as recalled next.

Definition 9.11. *Satisfiability in DL-Lite$_\mathcal{R}$ is* FOL-reducible *if, for every TBox $\mathcal{T}$ and RBox $\mathcal{R}$ expressed in DL-Lite$_\mathcal{R}$, there exists a Boolean FOL query q, over the alphabet of $\mathcal{T}$, such that for every nonempty ABox $\mathcal{A}$, $\langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle$ is satisfiable if and only if q evaluates to false in $db(\mathcal{A})$.*

Now, a query is constructed, by first splitting all GCI into those with $\neg$ on the right hand side (called *negative inclusions* (NI)) and those without (called *positive inclusions* (PI)) and by considering the NI as a starting point to compute all derivable NI.

Definition 9.12. *Let $\mathcal{T}$ be a DL-Lite$_\mathcal{R}$ TBox and $\mathcal{R}$ a DL-Lite$_\mathcal{R}$ RBox. We call* NI-closure *of $\mathcal{T}$ and $\mathcal{R}$, denoted by $cln(\mathcal{T}, \mathcal{R})$, the set defined inductively as follows:*

1. *All negative inclusion assertions in $\mathcal{T}$ and $\mathcal{R}$ are also in $cln(\mathcal{T}, \mathcal{R})$.*
2. *If $B_1 \sqsubseteq B_2$ is in $\mathcal{O}$ and $B_2 \sqsubseteq \neg B_3$ or $B_3 \sqsubseteq \neg B_2$ is in $cln(\mathcal{T}, \mathcal{R})$, then also $B_1 \sqsubseteq \neg B_3$ is in $cln(\mathcal{T}, \mathcal{R})$.*

Input: *DL-Lite*$_\mathcal{R}$ KB $\mathcal{O} = \langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle$
Output: true if $\mathcal{O}$ is satisfiable, false otherwise

$q_{unsat} = \bot$
for all $\alpha \in cln(\mathcal{T}, \mathcal{R})$ **do**
  $q_{unsat} = q_{unsat} \vee \delta(\alpha)$
end for
if $q_{unsat}^{db(\mathcal{A})} = \emptyset$ **then**
  return **true**
else
  return **false**
end if

Figure 9.1: Algorithm Consistent

3. *If* $R_1 \sqsubseteq R_2$ *is in* $\mathcal{O}$ *and* $\exists R_2 \sqsubseteq \neg B$ *or* $B \sqsubseteq \neg \exists R_2$ *is in* $cln(\mathcal{T}, \mathcal{R})$*, then also* $\exists R_1 \sqsubseteq \neg B$ *is in* $cln(\mathcal{T}, \mathcal{R})$*.*

4. *If* $R_1 \sqsubseteq R_2$ *is in* $\mathcal{O}$ *and* $\exists R_2^- \sqsubseteq \neg B$ *or* $B \sqsubseteq \neg \exists R_2^-$ *is in* $cln(\mathcal{T}, \mathcal{R})$*, then also* $\exists R_1^- \sqsubseteq \neg B$ *is in* $cln(\mathcal{T}, \mathcal{R})$*.*

5. *If* $R_1 \sqsubseteq R_2$ *is in* $\mathcal{O}$ *and* $R_2 \sqsubseteq \neg R_3$ *or* $R_3 \sqsubseteq \neg R_2$ *is in* $cln(\mathcal{T}, \mathcal{R})$*, then also* $R_1 \sqsubseteq \neg B$ *is in* $cln(\mathcal{T}, \mathcal{R})$*.*

6. *If one of the assertions* $\exists R \sqsubseteq \neg \exists R$*,* $\exists R^- \sqsubseteq \neg \exists R^-$*, or* $R \sqsubseteq \neg R$ *is in* $cln(\mathcal{T}, \mathcal{R})$*, then all three such assertions are in* $cln(\mathcal{T}, \mathcal{R})$*.*

Given the NI-closure of $\mathcal{T}$ and $\mathcal{R}$ we can translate each axiom into a first-order formula.

Definition 9.13. *Let* $\mathcal{O}$ *be a DL-Lite*$_\mathcal{R}$ *KB and* $cln(\mathcal{T}, \mathcal{R})$ *the NI-closure of* $\mathcal{T}$ *and* $\mathcal{R}$*. The translation function* δ *from axioms in* $cln(\mathcal{T}, \mathcal{R})$ *to first-order formulas is:*

$$\delta(B_1 \sqsubseteq \neg B_2) = \exists x.\gamma_1(x) \wedge \gamma_2(x)$$
$$\delta(R_1 \sqsubseteq \neg R_2) = \exists x, y.\rho_1(x, y) \wedge \rho_2(x, y)$$

where $\gamma_i(x) = A_i(x)$ *if* $B_i = A_i$*,* $\gamma_i(x) = \exists y_i.R_i(x, y_i)$ *if* $B_i = \exists R_i$*, and* $\gamma_i(x) = \exists y_i.R_i(y_i, x)$ *if* $B_i = \exists R^-$*; and* $\rho_i(x, y) = P_i(x, y)$ *if* $R_i = P_i$*, and* $\rho_i(x, y) = P_i(y, x)$ *if* $R_i = P_i^-$*.*

The algorithm in Fig. 9.1 checks satisfiability of a *DL-Lite*$_\mathcal{R}$ knowledge base $\mathcal{O}$. Of course, if $\mathcal{O}$ is free of NI, then the algorithm succeeds automatically.

With this, instance checking is straightforwardly obtained in [Calvanese et al., 2007].

Theorem 9.14. *Let $\mathcal{O}$ be DL-Lite$_\mathcal{R}$ KB, and either H a general concept (with ground argument t_i) appearing in $\mathcal{O}$ and $\hat{A}$ an atomic concept not appearing in $\mathcal{O}$ or H a role name or its inverse (with ground arguments t_i) appearing in $\mathcal{O}$ and $\hat{A}$ an atomic role not appearing in $\mathcal{O}$. Then $\mathcal{O} \models H(t_i)$ iff $\mathcal{O} \cup \{\hat{A} \sqsubseteq \neg H, \hat{A}(t_i)\}$ is unsatisfiable.*

Note that this theorem is a generalization of two separate theorems for concepts and roles.

9.2.2 Defining the Oracle for *DL-Lite*$_\mathcal{R}$

The material presented so far suffices to handle the bottom-up computation of the well-founded partition for hybrid MKNF knowledge bases. In the subsequent example, which we recall from [Calvanese et al., 2007], we not only present how satisfiability and instance checking work, but also consider possible solutions for defining an oracle.

Example 9.15. *Consider the DL-Lite$_\mathcal{R}$ KB $\mathcal{O}$ consisting of the axioms in the TBox and the RBox:*

$$\texttt{Professor} \sqsubseteq \exists \texttt{TeachesTo} \tag{9.1}$$

$$\texttt{Student} \sqsubseteq \exists \texttt{HasTutor} \tag{9.2}$$

$$\exists \texttt{TeachesTo}^- \sqsubseteq \texttt{Student} \tag{9.3}$$

$$\exists \texttt{HasTutor}^- \sqsubseteq \texttt{Professor} \tag{9.4}$$

$$\texttt{Professor} \sqsubseteq \neg \texttt{Student} \tag{9.5}$$

$$\texttt{HasTutor}^- \sqsubseteq \texttt{TeachesTo} \tag{9.6}$$

and the simple ABox:

$$\texttt{Student}(\texttt{John}) \quad \texttt{HasTutor}(\texttt{John}, \texttt{Mary}) \quad \texttt{TeachesTo}(\texttt{Mary}, \texttt{Bill}). \tag{9.7}$$

For checking satisfiability, we create $db(\mathcal{A})$ with the domain $\Delta^{db(\mathcal{A})} = \{\texttt{John}, \texttt{Mary}, \texttt{Bill}\}$ whose elements are all mapped to themselves, and the interpretation of all concept and role names according to $\mathcal{A}$, e.g., $\texttt{Student}^{\texttt{db}(\mathcal{A})} = \{\texttt{John}\}$ and $\texttt{Professor}^{\texttt{db}(\mathcal{A})} = \emptyset$.

Then, we calculate $cln(\mathcal{T},\mathcal{R})$ *as follows.*

$$\texttt{Professor} \sqsubseteq \neg\texttt{Student} \tag{9.8}$$

$$\exists\texttt{HasTutor}^- \sqsubseteq \neg\texttt{Student} \tag{9.9}$$

$$\exists\texttt{TeachesTo}^- \sqsubseteq \neg\texttt{Professor} \tag{9.10}$$

$$\exists\texttt{TeachesTo} \sqsubseteq \neg\texttt{Student} \tag{9.11}$$

$$\exists\texttt{HasTutor} \sqsubseteq \neg\texttt{Professor} \tag{9.12}$$

Axiom 9.8 occurs in $\mathcal{O}$*, Axiom 9.9 follows from (9.8) and (9.4), Axiom 9.10 follows from (9.8) and (9.3), and (9.11) and (9.12) follow from (9.6) and (9.9), respectively (9.10).*

The translation function δ *can be applied to each NI in* $cln(\mathcal{T},\mathcal{R})$*.*

$$\delta(\texttt{Professor} \sqsubseteq \neg\texttt{Student}) = \exists\texttt{x}.\texttt{Professor}(\texttt{x}) \wedge \texttt{Student}(\texttt{x}) \tag{9.13}$$

$$\delta(\texttt{HasTutor}^- \sqsubseteq \neg\texttt{Student}) = \exists\texttt{x}.(\exists\texttt{y}\texttt{HasTutor}(\texttt{y},\texttt{x})) \wedge \texttt{Student}(\texttt{x}) \tag{9.14}$$

$$\delta(\texttt{TeachesTo}^- \sqsubseteq \neg\texttt{Professor}) = \exists\texttt{x}.(\exists\texttt{y}\texttt{TeachesTo}(\texttt{y},\texttt{x})) \wedge \texttt{Professor}(\texttt{x}) \tag{9.15}$$

$$\delta(\texttt{TeachesTo} \sqsubseteq \neg\texttt{Student}) = \exists\texttt{x}.(\exists\texttt{y}\texttt{TeachesTo}(\texttt{x},\texttt{y})) \wedge \texttt{Student}(\texttt{x}) \tag{9.16}$$

$$\delta(\texttt{HasTutor} \sqsubseteq \neg\texttt{Professor}) = \exists\texttt{x}.(\exists\texttt{y}\texttt{HasTutor}(\texttt{x},\texttt{y})) \wedge \texttt{Professor}(\texttt{x}) \tag{9.17}$$

Considering $db(\mathcal{A})$ *and the disjunction of first-order formulas resulting from the translation yields a successful test for satisfiability.*

If we want to verify, e.g., $\texttt{Student}(\texttt{John})$*, then we extend* $\mathcal{O}$ *with* $\hat{\texttt{A}}(\texttt{John})$ *and* $\hat{\texttt{A}} \sqsubseteq \neg\texttt{Student}$ *resulting in* $\mathcal{O}'$*, update* $db(\mathcal{A}')$ *appropriately, and add three more NI to* $cln(\mathcal{T}',\mathcal{R}')$*:*

$$\hat{\texttt{A}} \sqsubseteq \neg\texttt{Student} \tag{9.18}$$

$$\exists\texttt{TeachesTo}^- \sqsubseteq \neg\hat{\texttt{A}} \tag{9.19}$$

$$\exists\texttt{HasTutor} \sqsubseteq \neg\hat{\texttt{A}} \tag{9.20}$$

These axioms can again be translated, and it can be easily verified that the resulting check yields unsatisfiability. From this, we derive that $\texttt{Student}(\texttt{John})$ *holds.*

$$\delta(\hat{\texttt{A}} \sqsubseteq \neg\texttt{Student}) = \exists\texttt{x}.\hat{\texttt{A}}(\texttt{x}) \wedge \texttt{Student}(\texttt{x}) \tag{9.21}$$

$$\delta(\texttt{TeachesTo}^- \sqsubseteq \neg\hat{\texttt{A}}) = \exists\texttt{x}.(\exists\texttt{y}\texttt{TeachesTo}(\texttt{y},\texttt{x})) \wedge \hat{\texttt{A}}(\texttt{x}) \tag{9.22}$$

$$\delta(\texttt{HasTutor} \sqsubseteq \neg\hat{\texttt{A}}) = \exists\texttt{x}.(\exists\texttt{y}\texttt{HasTutor}(\texttt{x},\texttt{y})) \wedge \hat{\texttt{A}}(\texttt{x}) \tag{9.23}$$

If we want to incorporate this into $\mathbf{SLG(\mathcal{O})}$*, then there are two possible outcomes. First, we may, e.g., query for* $\texttt{Student}(\texttt{John})$ *and the previously presented steps would*

derive this from $\mathcal{O}$ *alone, so that we would expect the empty answer for* $\mathbf{SLG}(\mathcal{O})$. *I.e., nothing needs to be added to* $\mathcal{O}$ *to derive the queried atom from* $\mathcal{O}$ *and a child* $\texttt{Student}(\texttt{John}) : - \mid$ *is created in the tree for* $\texttt{Student}(\texttt{John})$.

Alternatively, consider that the ABox is not present, but that, for simplicity, the corresponding statements occur as rule facts. In this case, we want the oracle to return a set of atoms, which if resolved prove the original query. Clearly, we can derive $\texttt{Student}(\texttt{John})$ *if the satisfiability test for* $\mathcal{O}$ *fails. This is the case if one of the disjuncts in* $q_{unsat}^{db(\mathcal{A})}$ *is satisfiable, e.g., if there is an* $\texttt{x}$ *such that* $\texttt{Professor}(\texttt{x}) \wedge \texttt{Student}(\texttt{x})$. *Of course, it is counterintuitive to prove that John is a student by showing that there is some other individual that is a professor and a student, i.e., by deriving some inconsistency in the interaction of* $\mathcal{O}$ *and the rules. Thus, all the disjuncts resulting from (9.13)–(9.17), do not yield meaningful derivations. Instead they yield derivations based on some general MKNF-inconsistency, which is not possible in a partial oracle (cf. Definition 8.28).*

However, if we resolve the disjuncts resulting from (9.21)–(9.23) with $\hat{\texttt{A}}(\texttt{John})$, *then we obtain meaningful answers that can be used in the tree of* $\texttt{Student}(\texttt{John})$. *Namely,* $\texttt{Student}(\texttt{John})$ *itself is obtained, which is then discarded in* $\mathbf{SLG}(\mathcal{O})$, *and* $(\exists \texttt{y}\texttt{TeachesTo}(\texttt{y}, \texttt{John}))$, *and* $(\exists \texttt{y}\texttt{HasTutor}(\texttt{John}, \texttt{y}))$ *are also obtained. In fact, we can introduce two children:*

$$\texttt{Student}(\texttt{John}) : - \mid \texttt{TeachesTo}(\texttt{y}, \texttt{John})$$

and

$$\texttt{Student}(\texttt{John}) : - \mid \texttt{TeachesTo}(\texttt{John}, \texttt{y})$$

to the tree with root $\texttt{Student}(\texttt{John})$ *and try to find appropriate proofs in the rules of a hybrid MKNF knowledge base* $\mathcal{K}$ *consisting of rules and a DL part in DL-Lite*$_\mathcal{R}$.

The insights gained with this example can be formalized in the algorithm (Fig. 9.2) that provides an oracle for *DL-Lite*$_\mathcal{R}$. We only have to formalize the resolution step of the newly introduced query atom with each of the results of applications of δ. The result of such a resolution step is either a ground (unary or binary) atom or a binary atom with one existentially quantified variable. To check whether adding this atom to $\mathcal{O}$ and the already derived information remains consistent, we additionally introduce a uniform notion that turns the new atom into DL notation.

Definition 9.16. *Let* $\mathcal{O}$ *be a DL-Lite*$_\mathcal{R}$ *KB,* α *an axiom in* $cln(\mathcal{T}, \mathcal{R})$, *and* $\delta(\alpha) = \exists \vec{x}.(C_1 \wedge C_2)$ *such that* H *is unifiable with mgu* θ *with* C_i, *for some* i, *in* $\delta(\alpha)$. *Then*

$res(\delta(\alpha), H)$ is defined as $(C_2)\theta$ if $i = 1$, and $(C_1)\theta$ otherwise. The DL representation $res_{DL}(\delta(\alpha), H)$ of $res(\delta(\alpha), H)$ is defined depending on the form of $res(\delta(\alpha), H)$:

$$res_{DL}(\delta(\alpha), H) = \begin{cases} res(\delta(\alpha), H) & \text{if } res(\delta(\alpha), H) \text{ is a ground atom} \\ (\exists R)(a) & \text{if } res(\delta(\alpha), H) = \exists y.R(a, y) \text{ for ground } a \\ (\exists R^-)(a) & \text{if } res(\delta(\alpha), H) = \exists y.R(y, a) \text{ for ground } a \end{cases}$$

We recall that assertions for complex concepts such as $(\exists R)(a)$ are represented by $A(a)$ and $A \sqsubseteq \exists R$ for a new concept name A. This encoding may also affect atoms appearing in $I^+_{\mathcal{F}_n}$ which is why we directly incorporate $I^+_{\mathcal{F}_n}$ into $\mathcal{O}$ in Fig. 9.2 to avoid a more complicated notation.

The algorithm itself proceeds as outlined in the example. It checks first whether $\mathcal{O}$ together with the already proven true knowledge yields a satisfiable knowledge base. If not, the algorithm stops and returns the empty set; thus, $\mathcal{O}$ is not used for further derivations. Otherwise, it proceeds with an instance check for the query, i.e., by checking for unsatisfiability of the extended knowledge base, and, in the case of success, returns a set containing only the empty answer, hence, an unconditional answer in the respective tree of **SLG($\mathcal{O}$)**. If the instance check fails, then, for all newly introduced axioms in $cln(\mathcal{T}', \mathcal{R})$, it is verified whether $res_{DL}(\delta(\alpha), \hat{A}(t_i))$ does not cause an inconsistency if added to $\mathcal{O}$ and the already derived knowledge. If this is successful then the corresponding atom $res(\delta(\alpha), \hat{A}(t_i))$ is included in the set of returned answers, which if proven true, allow us to derive the considered query.

We show that this algorithm provides a correct partial oracle for *DL-Lite*$_\mathcal{R}$ w.r.t. **SLG($\mathcal{O}$)**.

Theorem 9.17. *The algorithm DL-Lite$_\mathcal{R}$ Oracle is sound and complete, i.e., the returned answers in L correspond to the definition of a partial oracle for DL-Lite$_\mathcal{R}$ and the algorithm enables us to compute all the minimal sets L according to the partial oracle for DL-Lite$_\mathcal{R}$.*

Proof. We consider $\mathcal{O} \cup I^+_{\mathcal{F}_n} \cup L \models q$ where $\mathcal{O} \cup I^+_{\mathcal{F}_n} \cup L$ is consistent (Definition 8.28) and q is the queried atom.

We show soundness, i.e., we show that the returned answers in L correspond to the definition of a partial oracle for *DL-Lite*$_\mathcal{R}$.

If the algorithm returns $\mathcal{L} = \emptyset$, then $\mathcal{O} \cup I^+_{\mathcal{F}_n}$ is not consistent, which means that the result from the algorithm is sound for this case. Otherwise, if the algorithm returns

$\mathcal{L} = \{\emptyset\}$, then the instance check for the query succeeds and in this case (for consistent $\mathcal{O} \cup I^+_{\mathcal{F}_n}$ as checked) $\mathcal{O} \cup I^+_{\mathcal{F}_n} \models q$ holds, which is also a sound answer. Finally, if the algorithm returns $\mathcal{L} = \{L_1, \ldots, L_n\}$, then the direct instance check failed, but $\mathcal{O} \cup I^+_{\mathcal{F}_n} \cup L_i$ is consistent and $\mathcal{O} \cup I^+_{\mathcal{F}_n} \cup L_i \models q$ holds because the addition of L to $I^+_{\mathcal{F}_n}$ would exactly enable the instance check to succeed (see Theorem 9.14).

To show completeness, we have to show that the algorithm enables us to compute all the minimal sets L according to the partial oracle for $DL\text{-}Lite_{\mathcal{R}}$. First, if $\mathcal{O} \cup I^+_{\mathcal{F}_n}$ is not consistent, then the partial oracle does not return any answer and this is covered by the returned empty set $\mathcal{L}$ in the algorithm. Then, if $\mathcal{O} \cup I^+_{\mathcal{F}_n} \cup L \models q$ holds for empty L, then the only minimal answer for the partial oracle is the empty set. The algorithm $DL\text{-}Lite_{\mathcal{R}}$ Oracle returns exactly only the empty set. It remains to be shown that the correctness result holds for nonempty L as well. So suppose that L' is a nonempty minimal set such that $\mathcal{O} \cup I^+_{\mathcal{F}_n} \cup L' \models q$ and $\mathcal{O} \cup I^+_{\mathcal{F}_n} \cup L'$ is consistent. First, any minimal set can only consist of one atom due to the restricted syntax of GCI in $DL\text{-}Lite_{\mathcal{R}}$. Furthermore, joining $\mathcal{O} \cup I^+_{\mathcal{F}_n} \cup L'$ and $q = H(t_i)$ together with $\hat{A} \sqsubseteq \neg H$ and $\hat{A}(t_i)$ yields an inconsistent $DL\text{-}Lite_{\mathcal{R}}$ KB $\mathcal{O}_1$, hence a successful instance check for q (Theorem 9.14). If we remove L', then the KB is consistent but the check for consistency would still compute all axioms such that the boolean disjunctive query q_{uns} w.r.t. $\mathcal{O}_1$ without L' would be unsatisfiable as such, but satisfiable upon addition of L', i.e., indicate a successful instance check. This is exactly what the algorithm $DL\text{-}Lite_{\mathcal{R}}$ Oracle computes and, for consistent $\mathcal{O} \cup I^+_{\mathcal{F}_n} \cup L'$, L' is returned as one of the answers. Note that none of the $\alpha \in cln(\mathcal{T}, \mathcal{R})$ is considered since if one of these succeeds, the entire knowledge base is inconsistent. Thus, considering only $\alpha \in cln(\mathcal{T}', \mathcal{R}) \setminus cln(\mathcal{T}, \mathcal{R})$ suffices to find all possible sets L, since res is applicable by construction in each such case. □

The computational complexity of the algorithm follows from results in [Calvanese et al., 2007].

Theorem 9.18. *Let $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ be a hybrid MKNF knowledge base with $\mathcal{O}$ in $DL\text{-}Lite_{\mathcal{R}}$. An* **SLG($\mathcal{O}$)** *evaluation of a query q in the algorithm $DL\text{-}Lite_{\mathcal{R}}$ Oracle is decidable with combined complexity PTime.*

Proof. We know from the proof of Theorem 43 in Calvanese et al. [2007] that the combined complexity for computing the disjunctive formula using δ on α in $cln(\mathcal{T})$ is polynomial, while the evaluation w.r.t. $db(\mathcal{A})$ is in LogSpace. Consequently, instance checking and checking satisfiability for $DL\text{-}Lite_{\mathcal{R}}$ is in PTime. The algorithm $DL\text{-}Lite_{\mathcal{R}}$ Oracle applies one such satisfiability check for $\mathcal{O} \cup I_{\mathcal{F}_n}$ and conditionally a further one

for the instance check. Then, conditionally a set of (polynomially many in combined complexity) $\delta(\alpha)$ is processed (each in the worst case containing a further satisfiability check, which is a slight extension of the first). We conclude that the combined complexity of *DL-Lite*$_\mathcal{R}$ Oracle is in PTime. □

Intuitively, this result, which is more in the spirit of the approach of **SLG($\mathcal{O}$)** than the procedure devised for $\mathcal{REL}$, is achieved because GCI and RI are of a particular restricted form that only involves two (possibly negated) atoms each. So the oracle just needs to find single atoms as replies, and does not need to compute minimal subsets of arbitrary size in the power set of all atoms. Clearly, a similar approach that is not based on a translation of the DL into rules is much harder to achieve for $\mathcal{REL}$ simply because $\cap$ is allowed on the left hand side of GCI.

9.3 An Oracle for DLP

Providing an oracle for DLP is considerably easier than doing so for $\mathcal{REL}$ or *DL-Lite*$_\mathcal{R}$ simply because DLP is designed to be translatable into rules. Moreover, no classical negation occurs in the DL, and so inconsistencies cannot occur. Therefore, in this section, we limit ourselves to recalling the transformation from [Grosof et al., 2003] that turns a DLP DL knowledge base into rules. Then, given a hybrid MKNF knowledge base $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ we can combine the result of translating $\mathcal{O}$ with $\mathcal{P}$ and query the combined set of rules directly in SLG, i.e., **SLG($\mathcal{O}$)** without Oracle Resolution, where we also can avoid the doubling of $\mathcal{K}$ since inconsistencies and coherence are not relevant in this case.

We recall the mapping from [Grosof et al., 2003], starting with the translation of a GCI. Let A be in N_C, and C, D arbitrary concepts restricted as in Section 3.3.3, R in N_R, and x, y variables, with y being a new variable, i.e., one that has not previously been used. We define the mapping $\mathcal{T}$ for a GCI $C \sqsubseteq D$ as follows.

$\mathcal{T}(C \sqsubseteq D)$:	$Th(D, y) \leftarrow Tb(C, y)$
$Th(A, x)$:	$A(x)$
$Th((C \sqcap D), x)$:	$Th(C, x) \wedge Th(D, x)$
$Th((\forall R.C), x)$:	$Th(C, y) \leftarrow R(x, y)$
$Tb(A, x)$:	$A(x) \leftarrow$
$Tb((C \sqcap D), x)$:	$Tb(C, x) \wedge Tb(D, x)$
$Tb((C \sqcup D), x)$:	$Tb(C, x) \vee Tb(D, x)$
$Th((\exists R.C), x)$:	$R(x, y) \wedge Tb(C, y)$

As pointed out in [Grosof et al., 2003], resulting rules of the form $H \wedge H' \leftarrow B$ are translated into two rules $H \leftarrow B$ and $H' \leftarrow B$, resulting rules of the form $H \leftarrow (H' \leftarrow B)$ are translated into $H \leftarrow H' \wedge B$, and those of the form $H \leftarrow B \vee B'$ into two rules $H \leftarrow B$ and $H \leftarrow B'$.

For complex class expressions C, D that are allowed to appear on the left and the right hand side of a GCI we define the translation of $C \equiv D$, i.e., $\mathcal{T}(C \equiv D)$, as $\mathcal{T}(C \sqsubseteq D)$ and $\mathcal{T}(D \sqsubseteq C)$.

ABox statements and certain range restrictions are translated as follows.

$$\begin{array}{ll} \mathcal{T}(\top \sqsubseteq \forall R.D): & Th(D, y) \leftarrow R(x, y) \\ \mathcal{T}(\top \sqsubseteq \forall R^{-}.D): & Th(D, x) \leftarrow R(x, y) \\ \mathcal{T}(D(a)): & Th(D, a) \\ \mathcal{T}(R(a, b)): & R(a, b) \end{array}$$

Finally, axioms on roles are translated.

$$\begin{array}{ll} \mathcal{T}(R \sqsubseteq S): & S(x, y) \leftarrow R(x, y) \\ \mathcal{T}(R \equiv S): & \begin{cases} R(x, y) \leftarrow S(x, y) \\ S(x, y) \leftarrow R(x, y) \end{cases} \\ \mathcal{T}(R \equiv S^{-}): & \begin{cases} R(x, y) \leftarrow S(y, x) \\ S(x, y) \leftarrow R(y, x) \end{cases} \\ \mathcal{T}(\mathsf{Tra}(R)): & R(x, z) \leftarrow R(x, y) \wedge R(y, z) \end{array}$$

It is shown in [Grosof et al., 2003], that the mapping $\mathcal{T}$ is equivalence preserving. This allows us to show that we can use the translated DLP KB for querying.

Theorem 9.19. *Let $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ be a hybrid MKNF knowledge base with $\mathcal{O}$ in DLP. Then $\mathcal{K}_{\mathcal{DLP}} = (\emptyset, \mathcal{T}(\mathcal{O}) \cup \mathcal{P})$ is semantically equivalent to $\mathcal{K} = (\mathcal{O}, \mathcal{P})$.*

Proof. This result follows immediately from Theorem 1 of [Grosof et al., 2003], which shows the equivalence of $\mathcal{O}$ and $\mathcal{T}(\mathcal{O})$ and the same argument as for $\mathcal{REL}$, i.e., that we can integrate both rule sets directly to obtain the oracle to the DL part. □

The polynomial data complexity is thus also preserved.

Theorem 9.20. *Let $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ be a hybrid MKNF knowledge base with $\mathcal{O}$ in DLP. An* **SLG($\mathcal{O}$)** *evaluation of a query in $\mathcal{K}_{\mathcal{DLP}} = (\emptyset, \mathcal{T}(\mathcal{O}) \cup \mathcal{P})$ is decidable with data complexity PTime.*

Proof. The result is obtained as a special case from the general result (Theorem 8.30) because DLP is polynomial and the translation $\mathcal{T}(\mathcal{O})$ is linear. □

We have thus presented three concrete partial oracles that maintain the polynomial data complexity and that are based on different algorithmic ideas. For DLP, we simply take advantage of the existing translation into rules and use SLG alone for querying. The same is done eventually for $\mathcal{REL}$, but we apply the subsumption algorithm beforehand, and we remove certain axioms that would not yield any information w.r.t. queries. in the case of $DL\text{-}Lite_{\mathcal{R}}$, we use to slight extension of the algorithm for instance checking in $DL\text{-}Lite_{\mathcal{R}}$ to obtain an oracle that truly works as defined in Chapter 8. These three partial oracles serve as show cases that partial oracles as defined in Chapter 8 are, indeed, of practical use and maintain the polynomial data complexity.

Input: *DL-Lite*$_\mathcal{R}$ KB $\mathcal{O} = (\mathcal{T}, \mathcal{R}, \mathcal{A})$, which already contains $I^+_{\mathcal{F}_n}$, and a ground atomic query $q = H(t_i)$

Output: a set $\mathcal{L}$ of L_i such that $\mathcal{O} \cup L_i \models H(t_i)$ with $\mathcal{O} \cup L_i$ consistent

$\mathcal{L} = \emptyset$
$q_{unsat} = \bot$
for all $\alpha \in cln(\mathcal{T}, \mathcal{R})$ **do**
  $q_{unsat} = q_{unsat} \vee \delta(\alpha)$
end for
if $q_{unsat}^{db(\mathcal{A})} \neq \emptyset$ **then**
  $\mathcal{L} = \emptyset$
else
  $q_{inst} = q_{unsat}$
  $\mathcal{T}' = \mathcal{T} \cup \{\hat{A} \sqsubseteq \neg H\}$
  $\mathcal{A}' = \mathcal{A} \cup \{\hat{A}(t_i)\}$
  $\mathcal{O}' = (\mathcal{T}', \mathcal{R}, \mathcal{A}')$
  for all $\alpha \in cln(\mathcal{T}', \mathcal{R}) \setminus cln(\mathcal{T}, \mathcal{R})$ **do**
    $q_{inst} = q_{inst} \vee \delta(\alpha)$
  end for
  if $q_{inst}^{db(\mathcal{A}')} \neq \emptyset$ **then**
    $\mathcal{L} = \{\emptyset\}$
  else
    for all $\alpha \in cln(\mathcal{T}', \mathcal{R}) \setminus cln(\mathcal{T}, \mathcal{R})$ **do**
      $\mathcal{O}'' = \mathcal{O} \cup \{res_{DL}(\delta(\alpha), \hat{A}(t_i))\}$
      $q_{uns} = q_{unsat}$
      for all $\beta \in cln(\mathcal{T}'', \mathcal{R}) \setminus cln(\mathcal{T}, \mathcal{R})$ **do**
        $q_{uns} = q_{uns} \vee \delta(\beta)$
      end for
      if $q_{uns}^{db(\mathcal{A}'')} = \emptyset$ **then**
        $\mathcal{L} = \mathcal{L} \cup \{res(\delta(\alpha), \hat{A}(t_i))\}$
      end if
    end for
  end if
  return $\mathcal{L}$
end if

Figure 9.2: Algorithm *DL-Lite*$_\mathcal{R}$ Oracle

10

Conclusions

We have investigated tight combinations of open and closed world reasoning in the setting of combinations of DL and non-monotonic rules. In particular, we have studied the approach of hybrid MKNF knowledge bases [Motik and Rosati, 2010] and, based on that work, we have developed a new semantics that soundly approximates the former, but which provides a lower data complexity in general. We have provided two general reasoning algorithms – one to compute the entire model and one allowing us to query the combined knowledge base – and for the latter we provided procedures that are tailored to knowledge bases in which reasoning remains tractable. Thus, we have shown that combining open and closed world reasoning efficiently is possible and we have established that, among the approaches that provide similar computational complexity, our work is the one with significantly less restrictions on the expressiveness in general, and on the interaction between the DL and the rules. In particular, our approach is the only one among these that satisfies the four criteria established in Chapter 1.

In this chapter, we are going to recall the results in more detail (Section 10.1), before we point out possible lines of future work (Section 10.2).

10.1 Accomplishments

The achievements of this thesis can be split into three parts. Namely, the development of a new semantics; establishing an algorithm for computing the unique model of that semantics based on which the majority of the properties of the semantics can be proven;

and finally the definition of an accompanying top-down query algorithm, which promises to be scalable to very large knowledge bases in practice.

Based on the two-valued MKNF logics [Lifschitz, 1991], a three-valued semantics for MKNF formulas has been defined in Chapter 5, partially tailored (cf. the evaluation of the operator $\supset$) towards an integration of first-order DL and MKNF rules. This extension is faithful in the sense that any two-valued MKNF model is also a total three-valued MKNF model. For (non-disjunctive) hybrid MKNF knowledge bases, we have devised a special minimal model in this semantics, the well-founded MKNF model, which is the least among all three-valued MKNF models w.r.t. derivable knowledge. Our proposal straightforwardly satisfies the four criteria presented in Chapter 1 for the combination of rules and ontologies. Moreover, we have shown that if the knowledge base is consistent in our semantics,[1] then this model is unique and it exists. The proofs of these and other important properties are obtained based on the definition of the bottom-up computation of the unique model.

In Chapter 6, based on similar results in LP, we have defined an alternating fixpoint construction that allows us to compute a representation of the well-founded MKNF model. This computation ensures that if the knowledge base is consistent, then the approach is coherent in the sense of [Pereira and Alferes, 1992], i.e., if a formula φ is first-order false in the ontology, then the non-monotonic interpretation of φ in the rules is enforced to be false as well. Furthermore, if the knowledge base is inconsistent, then the proposed construction allows us to detect this without any substantial additional computational effort. Additionally, we have devised an alternative characterization based on unfounded sets (as known from logic programs) and shown that the calculated result of that characterization coincides with the one obtained from the alternating fixpoint construction.

In the same chapter, the following important properties have been shown to hold for the proposed semantics.

- The well-founded MKNF model is faithful w.r.t. the two-valued MKNF models of [Motik and Rosati, 2010], i.e., each query that is true (resp. false) in the well-founded MKNF model is also true (resp. false), in each two-valued MKNF model.

[1] Note that the set of inconsistent hybrid knowledge bases w.r.t. our new semantics is a strict subset of the hybrid MKNF knowledge bases that are inconsistent w.r.t. the two-valued MKNF semantics.

- Our proposal coincides with the original DL-semantics when no rules are present, and the original WFS of logic programs if the DL component is empty.

- The computational data complexity of our approach depends on the computational complexity of the applied DL, but it is in general in a strictly lower complexity class than the two-valued MKNF semantics, and if the considered DL is of polynomial data complexity, then the combination with rules remains tractable.

In Chapter 8, we have presented a general querying mechanism for hybrid MKNF knowledge bases, called **SLG**($\mathcal{O}$), that is sound and complete for our proposed semantics and sound for hybrid MKNF knowledge bases of [Motik and Rosati, 2010]. **SLG**($\mathcal{O}$) is an extension of SLG resolution with an oracle to query an accompanying first-order theory that accepts DL-safe conjunctive queries, returning all correct answer substitutions for variables in the query. The defined general procedure applies to any DL and under certain conditions maintains the data complexity of our new proposal. In Chapter 9, we also have provided concrete oracles for $\mathcal{REL}$, DLP and *DL-Lite*, i.e., the three tractable fragments that are underlying the OWL 2 profiles that are part of the W3C recommendations [Hitzler et al., 2009a] for the Semantic Web. These oracles are based on different techniques, either translating the DL fragment into rules (DLP and $\mathcal{REL}$) and/or manipulating existing reasoning algorithms so that the required information is available for **SLG**($\mathcal{O}$)(*DL-Lite*$_\mathcal{R}$ and $\mathcal{REL}$). We show that the oracles thus defined are correct with respect to the general procedure and maintain the polynomial (data) complexity.

10.2 Future Work

Several lines of future research can be considered and these fall into the three parts in which we achieved our results.

First, the three-valued extension of the two-valued MKNF logics [Lifschitz, 1991] is defined for arbitrary MKNF formulas, but then we focus on (non-disjunctive) hybrid MKNF knowledge bases when defining the well-founded MKNF model. One way of approximating our work with non-monotonic extensions of DL, such as [Donini et al., 2002], would be to study the well-founded MKNF model for more general MKNF formulas. However, it is not possible to extend this to arbitrary MKNF formulas

since the occurrence of disjunctions of **K**-atoms would immediately yield two minimal models, and so no least one among all three-valued MKNF models. Similar results could be achieved if we extend the recent embedding of monotonic DL-safe rules into DL using nominal schema [Krötzsch et al., 2011] to non-monotonic rules. In this case, the semantics of [Donini et al., 2002] might serve as a starting point, which could be appropriately adapted to embeddings of non-monotonic (MKNF) rules rather than permitting a more liberate usage of modal operators but then restricting it. This could then be generalized thus achieving an approximation of MKNF knowledge bases and the work of [Donini et al., 2002] but from the side of [Donini et al., 2002].

Another possible extension of our work can be achieved by lifting some of the restrictions on hybrid MKNF knowledge bases. As discussed before, introducing disjunctions in rules is not an option since it would in general create various minimal models. Extending the content of modal atoms to non-atomic formulas seems more viable, and, even though this can in principle be incorporated into knowledge bases by appropriate transformations into first-order logic, it would be interesting to provide, e.g., a semantics that allows the usage of classical negation in rules and that handles these directly without a preceding knowledge base transformation. This seems possible given that our semantics already deals with the impact of classical negation from the DL anyway, and further generalizations may follow. This may yield even more expressive combined languages, and it remains to be seen whether we can benefit from this in terms of computational complexity.

Instead of extending the syntax, future work may also consider the extension of the semantics to deal with inconsistencies. More concretely, another topic to be pursued is the definition of a paraconsistent version of the semantics defined in this thesis. It is worth noting that when inconsistencies come from the combination of rules and the DL-part, i.e., for inconsistent KBs with a consistent DL, the construction defined in this thesis already yields some results, e.g., in Example 6.27 we still derive that the CD `Bts` is interesting. This suggests that the method could be further exploited in the direction of defining a paraconsistent semantics for hybrid KBs, and, in $\mathbf{SLG}(\mathcal{O})$, due to its top-down definition, we also already have some paraconsistent behavior in the case of partial oracles.

The latter two lines of future work suggest to adapt $\mathbf{SLG}(\mathcal{O})$ to the proposed extensions, but there are many more open problems. Among the three tractable fragments

considered for concrete oracles, $\mathcal{REL}$ is the one that is considerably more restricted w.r.t. to the admissible language of the corresponding OWL 2 profile. Thus, we may extend the oracle for $\mathcal{REL}$ to $\mathcal{SROEL}$ [Baader et al., 2008] or even to its still tractable extensions ELP [Krötzsch, 2010; Krötzsch et al., 2008b] or even $\mathcal{SROELV}_n$ [Krötzsch et al., 2011].

Finally, we may consider implementations of the **SLG**($\mathcal{O}$). An implementation of this procedure that is based on XSB Prolog[1] for the tabling resolution is already part of the CVS version of XSB Prolog, and the description of this implementation can be found at [Gomes et al., 2010, 2011]. However, the DL considered in that implementation is the CDF framework already integrated in XSB, so that the approach is not applicable to arbitrary DL at all. Thus, we may develop an implementation that can be integrated with an arbitrary DL, e.g., as a plug-in for Protégé. Of special interest should be the implementation of the procedures tailored to tractable fragments, since this is where our semantics has its particular benefits in terms of data complexity. Consequently, we could turn the results of this thesis into practical systems that can be used in real applications.

[1] http://xsb.sourceforge.net/

Index

$\Gamma_{\mathcal{K}_G}$, 94
$\succeq_k$, 81

ABox, 44
admissible, DLs, 65
admissible, MKNF KB, 65
AEL, 137
$\mathcal{AL}$-log, 124
answer, 48
Answer Completion, 163
answer set, 31
answer, **SLG**($\mathcal{O}$), 159
atom, (first-order), 28

body, MKNF$^+$ rule, 60
body, rule, 29

CARIN, 125
closed, MKNF formula, 56
complete oracle, 162
complete, complexity, 28
completed $\mathcal{REL}$ KB, 180
completely evaluated, 159
Completion, 163
complexity, combined, 27
complexity, data, 27
complexity, taxonomic, 27
$compT_{\mathcal{O}}$, 162
concept subsumption, 47
concept unsatisfiability, 47
conditional answer, 159
conjunctive query, 48
conjunctive query, boolean, 48
coNP, 27
consistent, MKNF interpretation pair, 77
correct, partial oracle, 173
CWA, 7

$D_{\mathcal{K}_G^d}$, 150
$D_{\mathcal{K}_G}$, 92
decidable, decidability, 14
default negation, 29
Delaying, 163
depend on, 103
disjunctive dl-programs, 136
distinguished, 48
DL Rules, 127
$\mathcal{DL}$+log, 130
DL-atom, 64
DL-predicate, 64
dl-programs, 134
DL-safe, 64
DL-safe rules, 126

DLH, 52
DLP, 127
DLs, 5
doubled hybrid MKNF KB, 149
DP, 27

early evaluated, 160
ELP, 128
embeddings into AEL, 137
entailment, DLs, 46
entailment, MKNF, three-valued, 79
entailment, MKNF, two-valued, 58
equality, 56

f-hybrid knowledge bases, 133
fact, MKNF$^+$ rule, 60
fact, rule, 29
fail, child, 163
failed, atom, 161
failure node, 158
faithful, faithfulness, 13
flat, MKNF formula, 56
flexible, flexibility, 14
FOL-reducible, 185

g-hybrid knowledge bases, 132
GALEN, 10
GCI, 44
Gelfond-Lifschitz operator, 33
general concept inclusion axiom, 44
generalized atom, 60
generalized atom base, 60
GHB, 60
greatest fixpoint, 25
greatest unfounded set, 104, 152
ground instantiation, $\mathcal{K}$, 65
ground, MKNF formula, 56
grounding, 60

hard, complexity, 28
HB_Π, 30
head, MKNF$^+$ rule, 60
head, rule, 29
Herbrand base, 30
Herbrand universe, 30
HU_Π, 30
hybrid MKNF knowledge base, 67
hybrid MKNF$^+$ knowledge base, 61
hybrid rules with WFS, 138

inconsistency checking, 47
individual assertion, 44
induced, partial partition, 88
inequality, 56
instance checking, 47
integrity constraint, 60
interpretation induced by $\mathcal{F}$, 161
interpretation, DLs, 45
interpretation, LP, 31
interpretation, three-valued, LP, 34

K-atom, 56
$\mathsf{KA}(\mathcal{K}_G)$, 86
KB, 3
$\mathcal{K}_G//S$, MKNF coherent transform, 96
$\mathcal{K}_G/S$, MKNF transform, 94
$\mathcal{K}_G^d//'S$, MKNF^d coherent transform, 150
knowledge ordering, 35

KRR, 3

least fixpoint, 25
least model, 33
limit ordinal, 25
literal, 29
literal, negative, 29
literal, positive, 29
logic, two-valued, 31
logical consequence, 46
LogSpace, 27
LP, 7

MBNF, 55
MKNF, 17
MKNF interpretation, 58
MKNF interpretation pair, 77
MKNF knowledge base, 62
MKNF model, three-valued, 78
MKNF model, two-valued, 58
MKNF rule, 62, 67
MKNF satisfiable, 58
MKNF structure, 57
MKNF structure, three-valued, 75
MKNF transform, 94
MKNF unsatisfiable, 58
MKNF$^+$ atom, 60
MKNF^d-coherent transform, 150
MKNF-coherent transform, 96
MKNF-consistent, 79
MKNF-inconsistent, 79
modal atom, 56
modally closed, MKNF formula, 56
model, DL, 46
monotonic, 25

N2ExpTime, 27
name, 57
Negation Failure, 163
Negation Success, 163
negative delay literal, 158
negative inclusion, 185
Negative Return, 163
New Subgoal, 162
NExpTime, 27
NI, 185
$\mathbf{N}_i$, 99
NI-closure, 185
node, 158
nominal schema, 129
non-disjunctive, MKNF$^+$ rule, 60
non-distinguished, 48
non-DL-atom, 64
non-DL-predicate, 64
nonmodal atom, 60
normal form, $\mathcal{REL}$, 176
not-atom, 56
NP, 27

$\mathsf{OB}_{\mathcal{O},S}$, 87
objective knowledge, 87
ontology, 5
Oracle Resolution, 163
ordinal, 25
OWA, 4

partial oracle, 173
partial partition, 86

PI, 185
$\mathbf{P}_i$, 99
polynomial hierarchy, 28
positive delay literal, 158
positive inclusion, 185
Positive Return, 163
positive, MKNF formula, 56
positive, MKNF$^+$ rule, 60
Program Clause Resolution, 162
program tree, 158
program, definite, 30
program, disjunctive, 30
program, generalized, 30
program, generalized disjunctive, 29
program, MKNF$^+$, 60
program, normal, 30
program, positive, 30
$pT_{\mathcal{O}}$, 173
PTime, 27

QEL, 133
query answering, 48

$R_{\mathcal{K}_G^d}$, 150
$R_{\mathcal{K}_G}$, 92
RBox, 43
reduced $\mathcal{REL}$ KB, 180
residual program, 172
RIA, 41
role assertion, 43
role expression, 41
role hierarchy, 41
role hierarchy, regular, 41
role inclusion axiom, 41
root node for T, 158
rule, definite, 30
rule, disjunctive, 30
rule, generalized, 30
rule, normal, 30
rule, positive, 30
rules, generalized disjunctive, 29

safe, MKNF$^+$ rule, 60
safe, rule, 30
satisfiability, DL axioms, 46
satisfiability, DL KB, 46
satisfiability, interpretation pair, 77
satisfiability, LP rule, 31
satisfiability, MKNF, two-valued, 58
satisfiable, 46
set of **K**-atoms, 86
signature, 28
signature, DL, 40
signed set, 35
simple role, 42
Simplification, 163
SLG forest, 159
SLG resolvable, 159
SLG resolvent, 159
SMS, 8
SNOMED, 10
stable model, 34
standard name assumption, 59
standard reasoning task, 47
strict, MKNF formula, 56
subjective, MKNF formula, 56
successful, atom, 161

successor ordinal, 25
supported answer, 160
SWRL, 125

T_Π, definite logic program, 33
$T_{\mathcal{K}_G^d}$, 150
$T_{\mathcal{K}_G}$, 92
tabling, 157
TBox, 44
term, 28
three-valued model, LP, 35
tight, tightness, 13
total, MKNF interpretation pair, 77
total, MKNF structure, 75
T'_Π, normal logic program, 36
tractable, 28
transfinite induction, 26
transform, 31
tree for S, 159

underlying subgoal, 159
unfounded set, hybrid MKNF, 104, 152
unfounded set, LP, 36
unsatisfiable, 46
unsupported answer, 160

W3C, 3
well-founded MKNF model, 82
well-founded partition, 100
WFS, 8
WFS for dl-programs, 140
WFS for normal dl-programs, 141

References

João Alcântara, Carlos V. Damásio, and Luís M. Pereira. A well-founded semantics with disjunction. In Maurizio Gabbrielli and Gopal Gupta, editors, *Logic Programming, 21st International Conference, ICLP 2005, Sitges, Spain, October 2-5, 2005, Proceedings*, pages 341–355. Springer, 2005. 26

José J. Alferes, Matthias Knorr, and Terrance Swift. Queries to hybrid MKNF knowledge bases through oracular tabling. In Abraham Bernstein, David R. Karger, Tom Heath, Lee Feigenbaum, Diana Maynard, Enrico Motta, and Krishnaprasad Thirunarayan, editors, *The Semantic Web - ISWC 2009, 8th International Semantic Web Conference, ISWC 2009, Chantilly, VA, USA, October 25-29, 2009. Proceedings*, pages 1–16. Springer, 2009. 22

Krzysztof R. Apt and Roland N. Bol. Logic programming and negation: A survey. *Journal of Logic Programming*, 19/20:9–71, 1994. 7

Allesandro Artale, Diego Calvanese, Roman Kontchakov, and Michael Zakharyaschev. The *DL-Lite* family and relations. *Journal of Artificial Intelligence Research*, 36: 1–69, 2009. 54

Franz Baader. Least common subsumers and most specific concepts in a description logic with existential restrictions and terminological cycles. In Georg Gottlob and Toby Walsh, editors, *IJCAI-03, Proceedings of the 18th International Joint Conference on Artificial Intelligence*, pages 319–324. Morgan Kaufman, 2003. 50

Franz Baader and Bernhard Hollunder. $\mathcal{KRIS}$: $\mathcal{K}$nowledge $\mathcal{R}$epresentation and $\mathcal{I}$nference $\mathcal{S}$ystem. *SIGART Bull.*, 2(3):8–14, 1991. 6

Franz Baader and Bernhard Hollunder. Embedding defaults into terminological representation systems. *Journal of Automated Reasoning*, 14:149–180, 1995. 18, 122

Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the $\mathcal{EL}$ envelope. In Leslie P. Kaelbing and Alessandro Saffiotti, editors, *IJCAI-05, Proceedings of the 19th International Joint Conference on Artificial Intelligence*, pages 364–369. Morgan Kaufmann, 2005. 51, 180, 181, 182

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, 2nd edition, 2007a. 5, 7, 41, 63

Franz Baader, Baris Sertkaya, and Anni-Yasmin Turhan. Computing the least common subsumer w.r.t. a background terminology. *Journal of Applied Logic*, 5(3):392–420, 2007b. 50

Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the $\mathcal{EL}$ envelope further. In Kendell G. Clark and Peter F. Patel-Schneider, editors, *Proceedings of the OWLED 2008 DC Workshop on OWL: Experiences and Directions*, 2008. 51, 99, 205

Chitta Baral and Michael Gelfond. Logic programming and knowledge representation. *Journal of Logic Programming*, 19/20:73–148, 1994. 7

Chitta Baral, Jorge Lobo, and Jack Minker. Generalized well-founded semantics for logic programs. In M. E. Stickel, editor, *10th International Conference on Automated Deduction (CADE), Kaiserslautern, FRG, July 24-27, 1990, Proceedings*, pages 102–116. Springer, 1990. 26

Tim Berners-Lee, James Hendler, and Ora Lassila. The Semantic Web. *Scientific American*, pages 96–101, May 2001. 3

Harold Boley and Michael Kifer, editors. *RIF Basic Logic Dialect*. W3C Candidate Recommendation, 22 June 2010, 2010. Available at `http://www.w3.org/TR/rif-bld/`. 9

Harold Boley, Gary Hallmark, Michael Kifer, Adrian Paschke, Axel Polleres, and Dave Reynolds, editors. *RIF Core Dialect*. W3C Candidate Recommendation, 22 June 2010, 2010. Available at `http://www.w3.org/TR/rif-core/`. 9

Piero Bonatti, Carsten Lutz, and Frank Wolter. Expressive non-monotonic description logics based on circumscription. In Patrick Doherty, John Mylopoulos, and Christopher Welty, editors, *Proceedings of the Tenth International Conference on Principles of Knowledge Representation and Reasoning (KR'06)*, pages 400–410. AAAI Press, 2006. 18, 123

Piero A. Bonatti, Carsten Lutz, and Frank Wolter. The complexity of circumscription in DLs. *Journal of Artificial Intelligence Research (JAIR)*, 35:717–773, 2009. 18, 123

Ronald J. Brachman and Hector J. Levesque, editors. *Readings in Knowledge Representation*. Morgan Kaufman, Los Altos, 1985. 6, 218

Ronald J. Brachman and James G. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2):171–216, 1985. 6

Ronald J. Brachman, Deborah L. McGuinness, Peter F. Patel-Schneider, Lori A. Resnick, and Alexander Borgida. Living with CLASSIC: When and how to use a KL-ONE-like language. In John F. Sowa, editor, *Principles of Semantic Networks*, pages 401–456. Morgan Kaufman, Los Altos, 1991. 6

Stefan Brass and Jürgen Dix. Characterizations of the disjunctive well-founded semantics: Confluent calculi and iterated GCWA. *Journal of Automated Reasoning*, 20(1): 143–165, 1998. 26

Andrea Calì, Georg Gottlob, and Thomas Lukasiewicz. Datalog$^{\pm}$: a unified approach to ontologies and integrity constraints. In Ronald Fagin, editor, *Database Theory - ICDT 2009, 12th International Conference, St. Petersburg, Russia, March 23-25, 2009, Proceedings*, pages 14–30. ACM, 2009. 143

Diego Calvanese, Maurizio Lenzerini, and Daniele Nardi. Description logics for conceptual data modeling. In Jan Chomicki and Gunter Saake, editors, *Logics for databases and information systems*, chapter 8, pages 229–263. Kluwer Academic Publishers, 1998. 50

Diego Calvanese, Giuseppe de Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *Journal of Automated Reasoning*, 39(3):385–429, 2007. 53, 188, 189, 191, 195

Weidong Chen and David S. Warren. Tabled Evaluation with Delaying for General Logic Programs. *Journal of the ACM*, 43(1):20–74, 1996. 20, 150, 159, 162, 169, 170, 175, 177

Keith L. Clark. Negation as failure. In *Logic and Data Bases*, pages 293–322. Plenum Press, 1978. 8

Alain Colmerauer and Philippe Roussel. History of programming languages—ii. chapter The birth of Prolog, pages 331–367. ACM, New York, NY, USA, 1996. 8

Evgeny Dantsin, Thomas Eiter, Georg Gottlob, and Andrei Voronkov. Complexity and expressive power of logic programming. *ACM Computing Surveys*, 33(3):374–425, 2001. 19

Jos de Bruijn, Thomas Eiter, Axel Polleres, and Hans Tompits. Embedding non-ground logic programs into autoepistemic logic for knowledge-base combination. In Manuela M. Veloso, editor, *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI-07)*, pages 304–309, Hyderabad, India, January 6–12 2007a. AAAI Press. 17, 139

Jos de Bruijn, David Pearce, Axel Polleres, and Augustín Valverde. Quantified equilibrium logic and hybrid rules. In Massimo M. Marchiori, Jeff Z. Pan, and Christian de Sainte Marie, editors, *Proceedings of the First International Conference on Web Reasoning and Rule Systems*. Springer, 2007b. 17, 135

Jos de Bruijn, Thomas Eiter, Axel Polleres, and Hans Tompits. Embedding non-ground logic programs into autoepistemic logic for knowledge base combination. *ACM Transactions on Computational Logic*, 2011. accepted for publication. 17, 139

Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Andrea Schaerf. A hybrid system with datalog and concept languages. In Edoardo Ardizzone, Salvatore Gaglio, and Filippo Sorbello, editors, *2nd Congress of the Italian Association of Artificial Intelligence, AI∗IA Palermo, Italy, October, 29–31*, pages 88–97. Springer, 1991. 126

Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Andrea Schaerf. AL-log: Integrating datalog and description logics. *Journal of Intelligent Information Systems*, 10(3):227–252, 1998. 15, 126, 140

Francesco M. Donini, Daniele Nardi, and Riccardo Rosati. Description logics of minimal knowledge and negation as failure. *ACM Transactions on Computational Logic*, 3 (2):177–225, 2002. 17, 88, 123, 203, 204

Włodzimierz Drabent and Jan Małuszyński. Well-Founded semantics for hybrid rules. In Massimo M. Marchiori, Jeff Z. Pan, and Christian de Sainte Marie, editors, *Proceedings of the First International Conference on Web Reasoning and Rule Systems (RR2007)*, pages 1–15. Springer, 2007. 16, 122, 140

Włodzimierz Drabent and Jan Małuszyński. Hybrid rules with well-founded semantics. *Knowledge and Information Systems*, 25(1):137–168, 2010. 16, 122, 140, 141

Włodzimierz Drabent, Jakob Henriksson, and Jan Małuszyński. Hybrid reasoning with rules and constraints under well-founded semantics. In Massimo M. Marchiori, Jeff Z. Pan, and Christian de Sainte Marie, editors, *Proceedings of the First International Conference on Web Reasoning and Rule Systems (RR2007)*, pages 348–357. Springer, 2007. 141

Włodzimierz Drabent, Thomas Eiter, Giovambattista Ianni, Thomas Krennwallner, Thomas Lukasiewicz, and Jan Małuszyński. Semantic techniques for the web. chapter Hybrid reasoning with rules and ontologies, pages 1–49. Springer, Berlin, Heidelberg, 2009. 4, 15

Thomas Eiter, Thomas Lukasiewicz, Roman Schindlauer, and Hans Tompits. Well-Founded semantics for description logic programs in the semantic web. In Grigoris Antoniou and Harold Boley, editors, *Rules and Rule Markup Languages for the Semantic Web, RuleML'04*, pages 81–97. Springer, LNCS, 2004. 16, 122, 142

Thomas Eiter, Giovambattista Ianni, Roman Schindlauer, and Hans Tompits. Effective integration of declarative rules with external evaluations for semantic web reasoning. In York Sure and John Domingue, editors, *Proceedings of the 3rd European Conference on Semantic Web (ESWC 2006)*, pages 273–287. Springer, 2006. 16, 138

Thomas Eiter, Giovambattista Ianni, Thomas Krennwallner, and Axel Polleres. Rules and ontologies for the Semantic Web. In Cristina Baroglio, Piero A. Bonatti, Jan Małuszyński, Massimo Marchiori, Axel Polleres, and Sebastian Schaffert, editors,

Reasoning Web: 4th International Summer School 2008, Venice, Italy, September 7-11, 2008, Tutorial Lectures, pages 1–53. Springer, 2008a. 4, 15

Thomas Eiter, Giovambattista Ianni, Thomas Lukasiewicz, Roman Schindlauer, and Hans Tompits. Combining answer set programming with description logics for the Semantic Web. *Artificial Intelligence*, 172(12–13):1495–1539, August 2008b. 16, 17, 136, 137, 142

Thomas Eiter, Giovambattista Ianni, Thomas Lukasiewicz, and Roman Schindlauer. Well-founded semantics for description logic programs in the Semantic Web. *ACM Transactions on Computational Logic*, 12:11:1–11:41, January 2011. 16, 122, 142, 143

Cristina Feier and Stijn Heymans. Hybrid reasoning with forest logic programs. In Lora Aroyo, Paolo Traverso, Fabio Ciravegna, Philipp Cimiano, Tom Heath, Eero Hyvönen, Riichiro Mizoguchi, Eyal Oren, Marta Sabou, and Elena Paslaru Bontas Simperl, editors, *6th Annual European Semantic Web Conference (ESWC2009)*, pages 338–352, June 2009. 17, 135

Paolo Ferraris, Joohyung Lee, and Vladimir Lifschitz. Stable models and circumscription. *Artificial Intelligence*, 175:236–263, 2011. 123

Melvin Fitting. A Kripke-Kleene semantics for logic programs. *Journal of Logic Programming*, 4:295–312, 1985. 8

Melvin Fitting. *First-order logic and automated theorem proving*. Springer, 2nd edition, 1996. 33, 61

Melvin Fitting. Fixpoint semantics for logic programming — A survey. *Theoretical Computer Science*, 278(1–2):25–51, 2002. 37

Martin Gebser, Benjamin Kaufmann, André Neumann, and Torsten Schaub. clasp: A conflict-driven answer set solver. In Chitta Baral, Gerhard Brewka, and John S. Schlipf, editors, *Logic Programming and Nonmonotonic Reasoning, 9th International Conference, LPNMR 2007, Tempe, AZ, USA, May 15-17, 2007, Proceedings*, pages 260–265. Springer, 2007. 9

Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In Robert A. Kowalski and Kenneth A. Bowen, editors, *International Conference on Logic Programming, ICLP*, pages 1070–1080. MIT Press, 1988. 8, 19, 25, 34, 76, 93, 95, 96

Michael Gelfond and Vladimir Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9:365–385, 1991. 9, 25, 76

Birte Glimm, Carsten Lutz, Ian Horrocks, and Ulrike Sattler. Answering conjunctive queries in the $\mathcal{SHIQ}$ description logic. *Journal of Artificial Intelligence Research*, 31:150–197, 2008. 50

Ana S. Gomes, José J. Alferes, and Terrance Swift. Implementing query answering for hybrid MKNF knowledge bases. In Manuel Carro and Ricardo Peña, editors, *Practical Aspects of Declarative Languages, 12th International Symposium, PADL 2010, Madrid, Spain, January 18-19, 2010. Proceedings*, pages 25–39. Springer, 2010. 205

Ana Sofia Gomes, José J. Alferes, and Terrance Swift. A goal-directed implementation of query answering for hybrid MKNF knowledge bases. *Theory and Practice of Logic Programming*, 2011. accepted for publication, available at `http://arxiv.org/abs/1103.3949`. 205

Stephan Grimm and Pascal Hitzler. Semantic matchmaking of web resources with local closed-world reasoning. *International Journal of e-Commerce*, 12(2–Winter 2007–08): 89–126, 2008. 10, 18, 123

Stephan Grimm and Pascal Hitzler. A preferential tableaux calculus for circumscriptive ALCO. In Axel Polleres and Terrance Swift, editors, *Web Reasoning and Rule Systems, Third International Conference, RR 2009, Chantilly, VA, USA, October 20009, Proceedings*, pages 197–211. Springer, 2009. 18, 123

Stephan Grimm, Boris Motik, and Chris Preist. Matching semantic service descriptions with local closed-world reasoning. In York Sure and John Domingue, editors, *The Semantic Web: Research and Applications, 3rd European Semantic Web Conference, ESWC 2006, Budva, Montenegro, June 11–14, 2006, Proceedings*, pages 575–589. Springer, 2006. 18

Benjamin N. Grosof, Ian Horrocks, Raphael Volz, and Stefan Decker. Description logic programs: Combining logic programs with description logics. In *Proceedings of the World Wide Web Conference (WWW2003), Budapest, Hungary*, pages 48–57. ACM, 2003. 15, 54, 129, 187, 196, 197

Volker Haarslev, Kay Hidde, Ralf Möller, and Michael Wessel. The RacerPro knowledge representation and reasoning system. *Semantic Web journal*, 2011. to appear. Available at `http://www.semantic-web-journal.net/issues`. 6

Patrick J. Hayes. The logic of frames. In D. Metzing, editor, *Frame conceptions and text understanding*, pages 46–61. Walter de Gruyter and Co., 1979. Republished in Brachman and Levesque [1985]. 3, 5

Stijn Heymans, Davy Van Nieuwenborgh, and Dirk Vermeir. Open answer set programming for the Semantic Web. *Journal of Applied Logic*, 5(1):144–169, 2007. 17, 135

Stijn Heymans, Jos de Bruijn, Livia Predoiu, Cristina Feier, and Davy Van Nieuwenborgh. Guarded hybrid knowledge bases. *Theory and Practice of Logic Programming (TPLP), special issue Logic Programming and the Web*, 8(3):411–429, 2008a. 134, 135

Stijn Heymans, Davy Van Nieuwenborgh, and Dirk Vermeir. Open answer set programming with guarded programs. *ACM Transactions on Computational Logic*, 9 (4):1–53, 2008b. 134

Pascal Hitzler and Bijan Parsia. Ontologies and rules. In Steffen Staab and Rudi Studer, editors, *Handbook on Ontologies*, pages 111–132. Springer, 2 edition, 2009. 4, 15

Pascal Hitzler and Anthony K. Seda. *Mathematical Aspects of Logic Programming Semantics*. Studies in Informatics. Chapman and Hall/CRC Press, 2010. 26, 27

Pascal Hitzler and Matthias Wendt. A uniform approach to logic programming semantics. *Theory and Practice of Logic Programming*, 5(1–2):123–159, 2005. 87

Pascal Hitzler, Markus Krötzsch, Bijan Parsia, Peter F. Patel-Schneider, and Sebastian Rudolph, editors. *OWL 2 Web Ontology Language: Primer*. W3C Recommendation 27 October 2009, 2009a. Available at `http://www.w3.org/TR/owl2-primer/`. 4, 21, 179, 203

Pascal Hitzler, Markus Krötzsch, and Sebastian Rudolph. *Foundations of Semantic Web Technologies*. Chapman & Hall/CRC, 2009b. 5, 41

Matthew Horridge, Bijan Parsia, and Ulrike Sattler. Laconic and precise justifications in OWL. In Amit P. Sheth, Steffen Staab, Mike Dean, Massimo Paolucci, Diana Maynard, Timothy W. Finin, and Krishnaprasad Thirunarayan, editors, *The Semantic Web - ISWC 2008, 7th International Semantic Web Conference, ISWC 2008, Karlsruhe, Germany, October 26-30, 2008. Proceedings*, 2008. 50

Ian Horrocks and Peter F. Patel-Schneider. A proposal for an OWL rules language. In Stuart I. Feldman, Mike Uretsky, Marc Najork, and Craig E. Wills, editors, *Proceedings of the 13th International World Wide Web Conference (WWW 2004)*, pages 723–731. ACM, 2004. 13, 15, 127

Ian Horrocks, Peter F. Patel-Schneider, Harold Boley, Said Tabet, Benjamin N. Grosof, and Mike Dean, editors. *SWRL: A Semantic Web Rule Language*. W3C Member Submission, 21 May 2004, 2004. Available at `http://www.w3.org/Submission/SWRL/`. 127, 128

Ian Horrocks, Oliver Kutz, and Ulrike Sattler. The even more irresistible $\mathcal{SROIQ}$. In Patrick Doherty, John Mylopoulos, and Christopher A. Welty, editors, *Proceedings of the 10th International Conference on the Principles of Knowledge Representation and Reasoning (KR 2006)*, pages 57–67. AAAI Press, 2006a. 13, 41

Ian Horrocks, Boris Motik, Riccardo Rosati, and Ulrike Sattler. Can OWL and logic programming live together happily ever after? In Isabel F. Cruz, Stefan Decker, Dean Allemang, Chris Preist, Daniel Schwabe, Peter Mika, Michael Uschold, and Lora Aroyo, editors, *Proceedings of the 5th International Semantic Web Conference (ISWC 2006), volume 4273 of LNCS*, pages 501–514. Springer, 2006b. 18

Ulrich Hustadt, Boris Motik, and Ulrike Sattler. Data complexity of reasoning in very expressive description logics. In Leslie P. Kaelbling and Alessandro Saffiotti,

editors, *IJCAI-05, Proceedings of the 19th International Joint Conference on Artifical Intelligence*, pages 466–471. ACM, 2005. 15

Aditya Kalyanpur. Debugging and repair of OWL ontologies, 2006. Ph.D. Dissertation, University of Maryland College Park. 50

Yevgeny Kazakov. $\mathcal{RIQ}$ and $\mathcal{SROIQ}$ are harder than $\mathcal{SHOIQ}$. In Gerhard Brewka and Jérôme Lang, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Eleventh International Conference, KR 2008, Sydney, Australia, September 16-19, 2008*. AAAI Press, 2008. 50

Yevgeny Kazakov. Consequence-driven reasoning for Horn SHIQ ontologies. In Craig Boutilier, editor, *IJCAI-09, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009*, pages 2040–2045, 2009. 7

Peihong Ke and Ulrike Sattler. Next steps for description logics of minimal knowledge and negation as failure. In Franz Baader, Carsten Lutz, and Boris Motik, editors, *Proc. of the 2008 Description Logic Workshop (DL 2008)*, 2008. 18

Michael Kifer, Georg Lausen, and James Wu. Logical foundations of object-oriented and frame-based languages. *Journal of the ACM*, 42:741–843, July 1995. 9

Matthias Knorr and José J. Alferes. Querying in $\mathcal{EL}^+$ with nonmonotonic rules. In Helder Coelho, Rudi Studer, and Michael Wooldridge, editors, *Proceedings of the 19th European Conference on Artificial Intelligence, ECAI2010*, pages 1079–1080. IOS Press, 2010. 22

Matthias Knorr and Pascal Hitzler. A comparison of disjunctive well-founded semantics. In Pascal Hitzler, Thomas Roth-Berghofer, and Sebastian Rudolph, editors, *FAInt-07, Foundations of Artificial Intelligence, Workshop at KI 2007*. CEUR Workshop Proceedings, Vol. 277, 2007. 69

Matthias Knorr, José J. Alferes, and Pascal Hitzler. A well-founded semantics for MKNF knowledge bases. In Diego Calvanese, Enrico Franconi, Volker Haarslev, Domenico Lembo, Boris Motik, Anni-Yasmin Turhan, and Sergio Tessaris, editors, *Description Logics 2007*, pages 417–425. CEUR–WS, 2007a. 22

Matthias Knorr, José J. Alferes, and Pascal Hitzler. Towards tractable local closed world reasoning for the semantic web. In José Neves, Manuel F. Santos, and José Machado, editors, *Proceedings of the 13th Portuguese Conference on Artificial Intelligence, EPIA 2007, Guimarães, Portugal, December 3-7, 2007*, pages 3–14. Springer, 2007b. 22

Matthias Knorr, José J. Alferes, and Pascal Hitzler. A coherent well-founded model for hybrid MKNF knowledge bases. In Malik Ghallab, Constantine D. Spyropoulos, Nikos Fakotakis, and Nikos Avouris, editors, *Proceedings of the 18th European Conference on Artificial Intelligence, ECAI2008*, pages 99–103. IOS Press, 2008. 22, 81

Matthias Knorr, José J. Alferes, and Pascal Hitzler. Local closed world reasoning with description logics under the well-founded semantics. *Artificial Intelligence*, 175 (9–10):1528–1554, June 2011. 22

Kurt Konolige. Quantification in autoepistemic logic. *Fundamenta Informaticae*, 15: 275–300, 1991. 139

Robert A. Kowalski. The early years of logic programming. *Commun. ACM*, 31:38–43, January 1988. 7

Adila Alfa Krisnadhi, Frederick Maier, and Pascal Hitzler. OWL and rules. In *Reasoning Web 2011, Springer Lecture Notes in Computer Science*, 2011a. `http://knoesis.wright.edu/faculty/pascal/resources/publications/OWL-Rules-2011.pdf`, to appear. 4

Adila Alfa Krisnadhi, Kunal Sengupta, and Pascal Hitzler. Local closed world semantics: Keep it simple, stupid! Technical report, Wright State University, 2011b. available from `http://pascal-hitzler.de/resources/publications/GC-DLs.pdf`. 18, 123

M. Krötzsch, Frederick Maier, Adila A. Krisnadhi, and Pascal Hitzler. A better uncle for OWL: Nominal schemas for integrating rules and ontologies. In *Proceedings of the 20th International World Wide Web Conference, WWW2011, March/April 2011*, pages 645–654. ACM, 2011. 16, 131, 204, 205

REFERENCES

Markus Krötzsch. *Description Logic Rules*, volume 008 of *Studies on the Semantic Web*. IOS Press/AKA, 2010. 16, 41, 49, 51, 129, 130, 205

Markus Krötzsch, Sebastian Rudolph, and Pascal Hitzler. Conjunctive queries for a tractable fragment of OWL 1.1. In Karl Aberer, Key-Sun Choi, Natasha Noy, Dean Allemang, Kyung-Il Lee, Lyndon Nixon, Jennifer Golbeck, Peter Mika, Diana Maynard, Riichiro Mizoguchi, Guus Schreiber, and Philippe Cudré-Mauroux, editors, *Proceedings of the 6th International Semantic Web Conference (ISWC'07)*, volume 4825 of *LNCS*, pages 310–323. Springer, 2007. 52

Markus Krötzsch, Sebastian Rudolph, and Pascal Hitzler. Complexity boundaries for Horn description logics. In *Proceedings of the 22nd AAAI Conference on Artificial Intelligence (AAAI'07)*, pages 452–457. AAAI Press, 2007. 16

Markus Krötzsch, Sebastian Rudolph, and Pascal Hitzler. Description logic rules. In Malik Ghallab, Constantine D. Spyropoulos, Nikos Fakotakis, and Nikos Avouris, editors, *Proceedings of the 18th European Conference on Artificial Intelligence, ECAI2008*, pages 80–84. IOS Press, 2008a. 16, 129

Markus Krötzsch, Sebastian Rudolph, and Pascal Hitzler. ELP: Tractable rules for OWL 2. In Amit P. Sheth, Steffen Staab, Mike Dean, Massimo Paolucci, Diana Maynard, Timothy Finin, and Krishnaprasad Thirunarayan, editors, *Proceedings of the 7th International Semantic Web Conference (ISWC-08)*, pages 649–664. Springer, 2008b. 16, 129, 130, 205

Fritz Lehmann. *Semantic Networks in Artificial Intelligence*. Elsevier Science Inc., New York, NY, USA, 1992. 5

Nicola Leone, Gerald Pfeifer, Wolfgang Faber, Thomas Eiter, Georg Gottlob, Simona Perri, and Francesco Scarcello. The DLV system for knowledge representation and reasoning. *ACM Transactions on Computational Logic*, 7:499–562, July 2006. 9

Hector J. Levesque. Foundations of a functional approach to knowledge representation. *Artificial Intelligence*, 23:155–212, July 1984. 3

Alon Y. Levy and Marie-Christine Rousset. CARIN: A representation language combining Horn rules and description logics. In Wolfgang Wahlster, editor, *12th European Conference on Artificial Intelligence, Budapest, Hungary, August 11-16, 1996, Proceedings*, pages 323–327, 1996. 127

Alon Y. Levy and Marie-Christine Rousset. Combining Horn rules and description logics in CARIN. *Artificial Intelligence*, 104(1–2):165–209, 1998. 15, 127

Vladimir Lifschitz. Nonmonotonic databases and epistemic queries. In John Mylopoulos and Raymond Reiter, editors, *Proceedings of the 12th International Joint Conferences on Artifical Intelligence, IJCAI'91*, pages 381–386, 1991. 17, 19, 57, 60, 61, 64, 75, 76, 202, 203

Vladimir Lifschitz. Minimal belief and negation as failure. *Artificial Intelligence*, 70 (1–2):53–72, 1994. 57, 64, 65

John W. Lloyd. *Foundations of Logic Programming*. Springer, 2nd edition, 1987. 7, 35

Jorge Lobo, Jack Minker, and Arcot Rajasekar. *Foundations of disjunctive logic programming*. MIT Press, Cambridge, MA, USA, 1992. 25

Thomas Lukasiewicz. A novel combination of answer set programming with description logics for the Semantic Web. In Enrico Franconi, Michael Kifer, and Wolfgang May, editors, *Proceedings of the 4th European Semantic Web Conference (ESWC 2007)*, pages 384–398. Springer, 2007. 17, 138, 143

Thomas Lukasiewicz. A novel combination of answer set programming with description logics for the Semantic Web. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 22(11):1577–1592, November 2010. 17, 122, 138, 143

Carsten Lutz. The complexity of conjunctive query answering in expressive description logics. In Alessandro Armando, Peter Baumgartner, and Gilles Dowek, editors, *Proceedings of the 4th International Joint Conference on Automated Reasoning (IJCAR2008)*, number 5195 in LNAI, pages 179–193. Springer, 2008. 50

Robert MacGregor and Raymond Bates. The loom knowledge representation language. Technical Report ISI/RS-87-188, University of Southern California, Information Science Institute, Marina de Rey (CA, USA), 1987. 6

Tobias Matzner and Pascal Hitzler. Any-world access to OWL from Prolog. In Joachim Hertzberg, Michael Beetz, and Roman Englert, editors, *KI 2007: Advances in Artificial Intelligence, 30th Annual German Conference on AI, KI 2007, Osnabrück, Germany, September 2007, Proceedings*, pages 84–98. Springer, 2007. 17

John McCarthy. Circumscription – A form of non-monotonic reasoning. *Artificial Intelligence*, 13:27–39, 1980. 7, 123

Jing Mei, Shengping Liu, Guo Tong Xie, Aditya Kalyanpur, and Achille Fokoue. A practical approach for scalable conjunctive query answering on acyclic $\mathcal{EL}^+$ knowledge base. In Abraham Bernstein, David R. Karger, Tom Heath, Lee Feigenbaum, Diana Maynard, Enrico Motta, and Krishnaprasad Thirunarayan, editors, *The Semantic Web - ISWC 2009, 8th International Semantic Web Conference, ISWC 2009, Chantilly, VA, USA, October 25-29, 2009. Proceedings*, pages 408–423. Springer, 2009. 52

Jack Minker. On indefinite databases and the closed world assumption. In *Lecture Notes in Computer Science 138*, pages 292–308. Springer, Berlin, 1982. 25

Jack Minker and Dietmar Seipel. Disjunctive logic programming: A survey and assessment. In *Essays in Honour of Robert A. Kowalski, Part I, LNAI 2407*. Springer, 2002. 25

Marvin Minsky. A framework for representing knowledge. In John Haugeland, editor, *Mind Design: Philosophy, Psychology, Artificial Intelligence*, pages 95–128. MIT Press, Cambridge, MA, 1981. 5

Robert C. Moore. Semantical considerations on nonmonotonic logic. *Artificial Intelligence*, 25:75–94, 1985. 57, 139

Boris Motik and Riccardo Rosati. Closing semantic web ontologies. Technical report, University of Manchester, UK, 2006. URL `web.comlab.ox.ac.uk/people/Boris.Motik/pubs/mr06closing-report.pdf`. 82, 83

Boris Motik and Riccardo Rosati. A faithful integration of description logics with logic programming. In Manuela M. Veloso, editor, *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI-07)*, pages 477–482, Hyderabad, India, January 6–12 2007. AAAI Press. 58, 167

Boris Motik and Riccardo Rosati. Reconciling Description Logics and Rules. *Journal of the ACM*, 57(5):93–154, 2010. 13, 15, 17, 18, 19, 20, 21, 25, 57, 58, 61, 62, 64, 65, 66, 67, 68, 69, 76, 81, 82, 85, 87, 88, 89, 90, 94, 115, 117, 118, 122, 123, 125, 133, 137, 138, 139, 149, 174, 201, 202, 203

Boris Motik and Ulrike Sattler. A comparison of reasoning techniques for querying large description logic aboxes. In Miki Hermann and Andrei Voronkov, editors, *Proceedings of the 13th International Conference on Logic for Programming, Artificial Intelligencen, and Reasoning (LPAR01)*, pages 227–241. Springer, 2006. 6

Boris Motik, Ulrike Sattler, and Rudi Studer. Query-answering for OWL-DL with rules. *Journal of Web Semantics*, 3(1):41–60, 2005. 15, 66, 128, 129

Boris Motik, Bernardo Cuenca Grau, Ian Horrocks, Zhe Wu, Achille Fokoue, and Carsten Lutz, editors. *Profiles*. W3C Recommendation 27 October 2009, 2009a. Available at `http://www.w3.org/TR/owl2-profiles/`. 29, 51

Boris Motik, Peter F. Patel-Schneider, and Bijan Parsia, editors. *Structural specification and functional-style syntax*. W3C Recommendation 27 October 2009, 2009b. Available at `http://www.w3.org/TR/owl2-syntax/`. 44, 45

Boris Motik, Rob Shearer, and Ian Horrocks. Hypertableau Reasoning for Description Logics. *Journal of Artificial Intelligence Research*, 36:165–228, 2009c. 6

Bernhard Nebel and Kai von Luck. Hybrid reasoning in BACK. In *Proceedings of the 3rd International Symposium on Methodologies for Intelligent Systems (ISMIS'88)*, pages 260–269. North-Holland Publ. Co., Amsterdam, 1988. 6

Ilkka Niemelä and Patrik Simons. Smodels - an implementation of the stable model and well-founded semantics for normal LP. In *Proceedings of the 4th International Conference on Logic Programming and Nonmonotonic Reasoning*, LPNMR '97, pages 421–430, London, UK, 1997. Springer. 9

Christos H. Papadimitriou. *Computational Complexity*. Addison Wesley, 1994. 28, 29

Chintan Patel, James J. Cimino, Julian Dolby, Achille Fokoue, Aditya Kalyanpur, Aaron Kershenbaum, Li Ma, Edith Schonberg, and Kavitha Srinivas. Matching patient records to clinical trials using ontologies. In Karl Aberer, Key-Sun Choi,

Natasha Fridman Noy, Dean Allemang, Kyung-Il Lee, Lyndon J. B. Nixon, Jennifer Golbeck, Peter Mika, Diana Maynard, Riichiro Mizoguchi, Guus Schreiber, and Philippe Cudré-Mauroux, editors, *The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, November 11-15, 2007*, pages 816–829. Springer, 2007. 10, 149

Peter F. Patel-Schneider, Patrick Hayes, and Ian Horrocks, editors. *OWL Web Ontology Language Semantics and Abstract Syntax*. W3C Recommendation 10 Februar 2004, 2004. Available at `http://www.w3.org/TR/owl-semantics/`. 4

David Pearce and Augustín Valverde. A first order nonmonotonic extension of constructive logic. *Studia Logica*, 2–3(80):321–346, 2005. 135

Luís M. Pereira and José J. Alferes. Well founded semantics for logic programs with explicit negation. In Bernd Neumann, editor, *10th European Conference on Artificial Intelligence, ECAI 92, Vienna, Austria, August 3-7, 1992*, pages 102–106. John Wiley and Sons, Chichester, 1992. 20, 26, 98, 202

Antonella Poggi, Domenico Lembo, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. Linking data to ontologies. *Journal on Data Semantics*, X:133–173, 2008. 143

Teodor Przymusinski. On the declarative and procedural semantics of logic programs. *Journal of Automated Reasoning*, 5:167–205, 1989. 141

Teodor Przymusinski. Stable semantics for disjunctive programs. *New Generation Computing*, 9:401–424, 1991. 25

M. Ross Quillian. Word concepts: A theory and simulation of some basic capabilities. *Behavioral Science*, 12:410–430, 1967. 5

Raymond Reiter. On closed-world data bases. In *Logic and Data Bases*, pages 55–76. Plenum Press, 1978. 8

Raymond Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81–132, 1980. 7, 57, 122

Raymond Reiter. What should a database know? *Journal of Logic Programming*, 14 (1–2):127–153, 1992. 13

Riccardo Rosati. Reasoning about minimal belief and negation as failure. *Journal of Artificial Intelligence Research*, 11:277–300, 1999. 139

Riccardo Rosati. On the decidability and complexity of integrating ontologies and rules. *Journal of Web Semantics*, 3(1):41–60, 2005. ISSN 1570-8268. 17, 132

Riccardo Rosati. DL+Log: A tight integration of description logics and disjunctive datalog. In P. Doherty, J. Mylopoulos, and C. Welty, editors, *Tenth International Conference on the Principles of Knowledge Representation and Reasoning, KR'06*, pages 68–78. AAAI Press, 2006. 17, 132

Riccardo Rosati. On conjunctive query answering in EL. In Diego Calvanese, Enrico Franconi, Volker Haarslev, Domenico Lembo, Boris Motik, Anni-Yasmin Turhan, and Sergio Tessaris, editors, *Description Logics 2007*. CEUR Electronic Workshop Proceedings, 2007. 52

Kenneth A. Ross. The well founded semantics for disjunctive logic programs. In W. Kim, J.-M.. Nicolas, and S. Nishio, editors, *First International Conference on Deductive and Object Oriented Databases*, pages 385–402. Elsevier Science Publishers B.V. (North-Holland), 1990. 26

Francesca Rossi, Peter van Beek, and Toby Walsh. *Handbook of Constraint Programming (Foundations of Artificial Intelligence)*. Elsevier, New York, NY, USA, 2006. 9

Sebastian Rudolph, Markus Krötzsch, and Pascal Hitzler. Reasoning in $\mathcal{SHIQ}$ with ordered binary decision diagrams. In Dieter Fox and Carla P. Gomes, editors, *Proceedings of the 23rd AAAI Conference on Artificial Intelligence (AAAI08)*, pages 529–534. AAAI Press, 2008a. 7

Sebastian Rudolph, Markus Krötzsch, and Pascal Hitzler. Description logic reasoning with decision diagrams: Compiling $\mathcal{SHIQ}$ to disjunctive datalog. In Amit Sheth, Steffen Staab, Mike Dean, Massimo Paolucci, Diana Maynard, Timothy Finin, and Krishnaprasad Thirunarayan, editors, *Proceedings of the 7th International Semantic Web Conference (ISWC08)*, page 435450. Springer, 2008b. 7

Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 3rd edition, 2010. 3

K. Sagonas, T. Swift, and D. S. Warren. The limits of fixed-order computation. *Theoretical Computer Science*, 254(1-2):465–499, 2000. 163

Nigel Shadbolt, Tim Berners-Lee, and Wendy Hall. The Semantic Web revisited. *IEEE Intelligent Systems*, 21:96–101, May 2006. available at `http://dx.doi.org/10.1109/MIS.2006.62`. 3

Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. Pellet: A practical OWL-DL reasoner. *Web Semantics*, 5:51–53, 2007. 6

Terrance Swift. A new formulation of tabled resolution with delay. In *Recent Advances in Artifiial Intelligence*, volume 1695 of *LNAI*, pages 163–177. Springer, 1999. 159

Terrance Swift, Alexandre M. Pinto, and Luís M. Pereira. Incremental answer completion. In Patricia M. Hill and David Scott Warren, editors, *Logic Programming, 25th International Conference, ICLP 2009, Pasadena, CA, USA, July 14-17, 2009. Proceedings*, pages 519–524, 2009. 163

Alfred Tarski. Lattice-theoretic fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5(2):285–309, 1955. 27

Dmitry Tsarkov and Ian Horrocks. Fact++ description logic reasoner: System description. In Ulrich Furbach and Natarajan Shankar, editors, *In Proceedings of the International Joint Conference on Automated Reasoning (IJCAR 2006)*, pages 292–297. Springer, 2006. 6

Maarten H. van Emden and Robert A. Kowalski. The semantics of predicate logic as a programming language. *Journal of ACM*, 23:733–742, 1976. 8, 34

Allen van Gelder. The alternating fixpoint of logic programs with negation. In *Principles of Database Systems*, pages 1–10. ACM Press, 1989. 9, 25, 39, 40, 87, 93, 96, 98

Allen van Gelder, Kenneth A. Ross, and John S. Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM*, 38(3):620–650, 1991. 9, 19, 26, 36, 38, 39, 69, 76

Moshe Y. Vardi. Why is modal logic so robustly decidable? In Neil Immerman and Phokion G. Kolaitis, editors, *Descriptive Complexity and Finite Models, Proceedings of a DIMACS Workshop, January 14-17, 1996, Princeton University*, pages 149–184. American Mathematical Society, 1996. 13

Kewen Wang. A comparative study of well-founded semantics for disjunctive logic programs. In Thomas Eiter, Wolfgang Faber, and Miroslaw Truszczynski, editors, *Proceedings of the 6th International Conference on Logic Programming and Non-monotonic Reasoning*, LPNMR '01, pages 133–146. Springer, 2001. 26

Yisong Wang, Jia-Huai You, Li Yan Yuan, and Yi-Dong Shen. Loop formulas for description logic programs. *Theory and Practice of Logic Programming*, 10(4–6): 531–545, 2010. 137