Syntax-Preserving Belief Change Operators for Logic Programs and Hybrid Knowledge Bases

by

Sebastian Binnewies

MInfTech (Hons), Griffith University, Australia

Submitted in fulfilment of
the requirements of the degree of
Doctor of Philosophy

School of Information and Communication Technology (ICT)
Griffith Sciences
Griffith University, Australia

December 2015

Abstract

The recent years have seen several proposals aimed at placing the revision of logic programs within the belief change frameworks established for classical logic. A crucial challenge of this task lies in the nonmonotonicity of standard logic programming semantics. Existing approaches have thus used the monotonic characterisation via strong equivalence models to develop semantic revision operators, which however neglect any syntactic information. In this thesis, we bridge the gap between semantic and syntactic techniques by adapting three different types of constructions from classic belief change. Not only do they allow us to define new model-based revision operators that preserve the structure of the programs involved, but they also facilitate a natural definition of contraction operators for logic programs. In particular, we introduce partial meet revision and contraction operators, ensconcement revision and contraction operators, and entrenchment revision and contraction operators for logic programs. We present a new translation of the AGM and belief base revision and contraction postulates to logic programs that is closer to the original formulation than existing translations. Our partial meet and ensconcement operators conform to the belief base framework and our entrenchment operators are well-behaved with respect to the basic AGM postulates. We inter-define revision and contraction for each construction and show that ensconcement revision and contraction are generalisations of their partial meet counterparts. We also confirm that ensconcements may be indiscriminately defined over rules or subsets of a program without affecting the result of a revision or contraction. To increase efficiency of our constructions, we develop an algorithm that relies on modularising a logic program to reduce the revision or contraction of a logic program to performing revision or contraction only on the relevant modules of that program. We further define entrenchments as orderings only over the programs involved in the change operation rather than over the whole language. In relation to existing work, we compare our

approach to two state-of-the-art logic program revision methods and demonstrate that our operators address the shortcomings of one and generalise the other method. Based on our results for logic programs, we extend our operators to hybrid knowledge bases, which integrate logic programs and first-order theories in a single formalism, and compare belief change operations under two different hybrid knowledge base semantics.

Statement of Originality

This work has not previously been submitted for a degree or diploma in any university. To the best of my knowledge and belief, the report contains no material previously published or written by another person except where due reference is made in the thesis itself.

Signed:

_____

Sebastian Binnewies
December 2015

Acknowledgements

It has been quite a journey! Over rolling hills, across deserts, through jungles, and along shores to the horizon, with detours, dead-ends, milestones, and destinations. Now that I have arrived, I am looking back and wish to thank...

... my parents for equipping me with the right mindset to embark on this quest.

... Kewen Wang for providing me the compass that would guide me in the right direction during the many stages of this trip.

... Bela Stantic for all his advice as a seasoned traveller on these paths that I took for the first time.

... Zhiqiang Zhuang for being passionate and critical about my discoveries I made along the way.

... Yisong Wang for helping me climb the first mountain.

... Zhe Wang for the useful directory to stop me from getting lost.

... Irene Blee for giving me extra motivation to complete my journey with vigour.

... Lukas Folkman and the members of the AIST research group for being fellow travellers and sharing our experiences.

... Abdul Sattar for helping me venture beyond my known shores.

... Vicky Wheeler and Natalie Dunstan for captaining a ship whenever I had to cross waters.

... John Thornton for igniting the dreams that led to this journey.

And I am especially grateful to my wife Hiroko for being by my side every step of the way.

List of Publications

Publications

1. Binnewies, S., Wang, Y., Stantic, B., & Wang, K. (2013), Rule Revision in Normal DL Logic Programs, in 'Web Reasoning and Rule Systems', Vol. 7994 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 204–209.

2. Binnewies, S., Zhuang, Z., & Wang, K. (2015), Partial Meet Revision and Contraction in Logic Programs, *in* 'Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, AAAI 2015', AAAI Press, pp. 1439–1445.

Contents

List of Figures

List of Tables

1

Introduction

A key ingredient for any machine to be considered 'artificially intelligent' is a system to represent knowledge (McCarthy 1958). In analogy to a human brain, a knowledge representation system should be capable of storing information and reasoning over existing information to deduce new information. Moreover, information in a knowledge representation system should be amenable to change, whether it be adding, deleting, or modifying information in the system. The study of belief change (Doyle 1979, Fagin, Ullman & Vardi 1983, Gärdenfors 1988, Hansson 1999, Harper 1976, Levi 1980) concerns itself exactly with these kinds of dynamics in knowledge representation systems. It aims at providing mechanisms to change a knowledge representation system whenever new information is acquired. The majority of these mechanisms rely on two fundamental principles: the principle of primacy of new information, stating that information new to the system should be treated with priority over existing information in the system's knowledge base, and the principle of minimal change, stating that as much existing information as possible should be preserved during a change operation (Dalal 1988). We can illustrate the motivation for the two principles by considering the situation encountered by Dorothy in "The Wonderful Wizard of Oz" by L. Frank Baum.

> When Dorothy finds herself in the land of Oz, she believes that the only way to return home is by a magical spell and that a wizard by the name Oz can cast this spell. Later she discovers that Oz is actually not a wizard.

In Dorothy's case, the discovery that Oz is not a wizard is new information that she believes in. However, since this new information is inconsistent with her previous beliefs, she has to make up her mind. She discards her previous belief that Oz is a wizard to arrive at a consistent state. Note that she only discards her belief that Oz is a wizard, which implies to her that Oz cannot cast spells, but she does not discard her belief that she can return home by a magical spell in general, reflecting the principle of minimal change. This is indeed very beneficial to her, because if she gave up the latter belief as well, she would not be able to return home eventually via the magic spell from the silver shoes.

An important endeavour to guide change operations on a knowledge base and by now the most widely-adopted belief change paradigm is the so-called *AGM framework*, named after the initials of the author trio (Alchourrón, Gärdenfors & Makinson 1985). It classifies the possible changes to a knowledge base as *expansion*, *revision*, and *contraction* operations. In an expansion, new information is incorporated into a knowledge base, regardless of any inconsistencies that may arise. A revision operation also incorporates new information into a knowledge base, but in such a way that the resulting knowledge base is consistent. This is achieved by discarding some existing information. During a contraction, no new information is added to a knowledge base but some existing information is removed from it. On the one hand, the framework provides a set of postulates that each rational change operator should satisfy, and, on the other hand, defines specific constructions of expansion, revision, and contraction that satisfy these criteria. While the underlying assumption of the AGM framework is that any information implied by a knowledge base is represented explicitly in the knowledge base, the *belief base framework* of belief change (Fuhrmann 1991, Hansson 1989, Rott 1992) does not require this assumption. Postulates and constructions for expansion, revision, and contraction operators in the belief base framework have been defined to complement those from the AGM model (Hansson (1999) provides a summary).

While the AGM and belief base frameworks have been applied to a variety of knowledge representation formalisms (an overview is given by Wassermann

(2011)), work on an adaptation to knowledge representation in the form of *logic programs* has been slow to progress. Logic programming (Colmerauer & Roussel 1996, Kowalski 1974, Lloyd 1987), is a programming approach geared towards declarative, rather than procedural, problem-solving with an expressive, yet natural representation format. It forms the foundation of answer set programming (Gelfond & Lifschitz 1988, Baral 2003), which has been used for practical applications such as work-force management (Brewka, Eiter & Truszczyński 2011), molecular biology (Gebser, Guziolowski, Ivanchev, Schaub, Siegel, Thiele & Veber 2010), e-Tourism (Grasso, Leone & Ricca 2013), or linguistics (Brooks, Erdem, Minett & Ringe 2005).

A major challenge in the adaptation of the AGM and belief base frameworks to logic programming lies in the semantics of logic programs. While the frameworks and their previous adaptations are based on monotonic semantics, the standard answer set semantics of logic programs is nonmonotonic. Only recently have operators been proposed for belief revision in logic programs. Program-level revision (Delgrande 2010) and belief base revision (Krümpelmann & Kern-Isberner 2012) are initial approaches to logic program revision, yet have strict limitations in their expressiveness due to the nonmonotonicity of the underlying answer set semantics. A breakthrough arrived with the distance-based approach (Delgrande, Schaub, Tompits & Woltran 2013) to logic program revision, which rests upon characterising an agent's beliefs in terms of the set of *SE (strong equivalence) models* (Lifschitz, Pearce & Valverde 2001, Turner 2003) of a logic program. A logic program P is strongly equivalent to a program Q iff, for any program R, the answer sets of P combined with R are exactly the same as the answer sets of Q combined with R. SE model semantics provides a monotonic characterisation for logic programs and thus circumvents any obstacles presented by nonmonotonicity. To revise a program P by a program Q, the distance-based revision operator determines those SE models from the set of SE models of Q that are closest to the SE models of P.

1.1 Motivation

Even though the distance-based approach is considered a major milestone for logic program revision, it has some critical shortcomings. Firstly, as it relies on the set of SE models of an entire program as the representation of beliefs

expressed by the program, it operates on the *program-level* only. This means that a program may freely be substituted with any other that has the same set of SE models and the revision output will remain the same. However, the information expressed by a program is more than just its set of SE models – a program also encodes relationships between the atoms occurring in it (Leite & Pereira 1998). Such relationships are not adequately represented by the set of SE models of a program, particularly for the purpose of revising a program (Inoue & Sakama 2004). The consequence is that the distance-based approach essentially neglects information expressed on the *rule-level*, which leads to some highly unintuitive results. Consider the following two real-life examples.

> It is the 31st of December and I plan to drive from San Jose to San Francisco to see the New Year fireworks. Due to previous experience I believe that if there is heavy fog in San Francisco, then the city will cancel the fireworks. It has been clear and sunny for the last days, so I believe that it will not be foggy today either. I decide to check the weather forecast nonetheless, which says that there will be heavy fog tonight in San Francisco. Since I trust the forecast more than my own meteorological skills, I have to revise my beliefs. By employing the distance-based revision method, I would end up believing that it will be foggy, while being undecided whether the fireworks will be cancelled.

> I drive from San Jose to San Francisco every morning for work. I can use the 101 highway or the 280 freeway, but neither is particularly quicker. However, I believe that if there are roadworks on the 101, then the 280 is quicker. I was told by a friend that there are roadworks on the 101 currently, so I have been travelling on the 280. Now I hear on the radio that the roadworks finished and revise my beliefs. Using the distance-based revision method, I would end up with the belief that there are no roadworks on the 101, while still keeping the belief that the 280 is quicker.

In the first example, it is not clear why I do not conclude that the fireworks will be cancelled. The revision operation simply disregards that there exists a relationship between fog and fireworks cancellation. In the second example, I keep believing that the 280 is quicker, although the grounds to believe so do not hold any longer. We will formalise these two examples later after

we have introduced the required notation. A second shortcoming is that the distance-based approach makes the definition of a corresponding contraction operator difficult to come by. Firstly, in classical logic contraction can be defined in terms of revision by using the negation of a sentence. However, in logic programs we do not have the luxury of negation of a program. A workaround could be to use the complement of the set of SE models of the contracting program Q and select from this set the SE models that are closest to the ones of the initial program P. Yet, such a method may return SE models that are somewhat unrelated to P or Q, particularly when the complement consists of a large number of SE models.

The motivation for this work is to address these limitations. In particular, we aim to produce, on the one hand, revision operators that take into account information expressed by a program on the program-level and the rule-level in order to avoid such unintuitive results as just shown. On the other hand, we aim to provide corresponding contraction operators with similar properties. To ensure the rationality of our operators, we evaluate them against adaptations of the postulates from the AGM and belief base frameworks.

Additionally, we will investigate how to perform belief change when nonmonotonic logic programs are combined with monotonic description logics into a *hybrid knowledge base*. With a growing acceptance of description logics to model information in biomedical domains, addressing the dynamics of such information is of particular practical relevance. We aim to present rational operators to perform changes in hybrid knowledge bases. The operators should distinguish between changes to the nonmonotonic logic program component and the monotonic description logic component.

1.2 Approach

To achieve our aim of respecting information on the program- and rule-level during a belief change operation on logic programs, we first adapt the idea of a partial meet construction from classic belief change. This type of construction allows us to define model-based operators for revising as well as contracting logic programs while preserving their syntactic structure. We then focus on different constructions for logic program belief change that are based on ordering the beliefs expressed by a program by their relative importance. To do so, we translate the notions of ensconcements and entrenchments to logic

programs and introduce belief change operators that are determined by an ensconcement or entrenchment ordering associated with a program.

We provide a new translation of the AGM and belief base revision and contraction postulates and test our operators against them. We compare the properties of our operators to each other and establish formal relationships wherever suitable. To increase the efficiency of our constructions, we develop an algorithm to optimise our revision and contraction operations. We further investigate how our revision operators address the limitations of the distance-based approach (Delgrande, Schaub, Tompits & Woltran 2013) and whether they align closer to the AGM and belief base frameworks than the latter.

Based on the results we obtain for belief change in logic programs, we explore how they can be extended to hybrid knowledge bases. We lift our revision and contraction postulates to hybrid knowledge bases, define belief change operators under two different semantics for hybrid knowledge bases, and compare their properties.

1.3 Structure of the Thesis

- In Chapter 2, we cover the formal preliminaries required for the following chapters.
- In Chapter 3, we review existing work that is most relevant to this thesis.
- In Chapter 4, we present new translations of the AGM and belief base revision and contraction postulates to the logic program setting and establish formal relationships between these postulates and to previous translations. We also extend the postulates to cover belief change in hybrid knowledge bases.
- In Chapter 5, we introduce new revision and contraction operators for logic programs based on a partial meet construction from classic belief change. We provide representation theorems for partial meet revision and contraction with respect to the belief base postulates and show that partial meet revision can be characterised in terms of partial meet contraction and vice versa. In addition, we present an algorithm to optimise the operation of revising or contracting a logic program.

- In Chapter 6, we adapt the concepts of ensconcements and entrenchments from classic belief change to logic programs. We define revision and contraction operators based on an ensconcement or entrenchment associated with a logic program. We show that our ensconcement operators comply with the belief base framework and our entrenchment operators are well-behaved with respect to the basic AGM postulates. We demonstrate that ensconcement revision can be characterised in terms of ensconcement contraction and vice versa, establish that our ensconcement operators are generalisations of our partial meet operators, confirm that the outcome of a revision or contraction operation remains unaffected whether an ensconcement is defined over rules or subsets of a program, and show that the algorithm from the previous chapter is also applicable to compute ensconcement operations.

- In Chapter 7, we connect our results to the classic belief change frameworks and show which of our operators possess similar properties as their counterparts in propositional logic.

- In Chapter 8, we demonstrate that our partial meet and ensconcement revision operators address the shortcomings of the distance-based approach to logic program revision (Delgrande, Schaub, Tompits & Woltran 2013) and exhibit better properties with respect to the AGM and belief base frameworks, and that they are generalisations of the screened semi-revision approach for logic programs (Krümpelmann & Kern-Isberner 2012).

- In Chapter 9, we define belief change operators for hybrid knowledge bases. We first look at normal DL logic programs, which integrate a logic program with a description logic knowledge base under extended answer set semantics. We develop a new semantics for normal DL logic programs and show that it generalises the standard semantics for normal DL logic programs. We define a revision operator for normal DL logic programs and evaluate it against the extended belief base revision postulates. We then consider hybrid knowledge bases under extended here-and-there semantics, which generalise the integration with a logic program to any first-order theory. We adapt our partial meet revision and contraction operators for logic programs to capture changes to either the logic program component, the theory component, or both. We give representation theorems for partial meet revision and contraction in hy-

brid knowledge bases with respect to the extended belief base postulates.

- In Chapter 10, we conclude the thesis with a summary of its contributions and an outlook on future work.

Preliminary results from Chapters 5 and 8 are published as (Binnewies, Zhuang & Wang 2015). Parts of Chapter 9 are published as (Binnewies, Wang, Stantic & Wang 2013).

2

Preliminaries

To provide the preliminaries required for the following chapters, here we first recall syntax and semantics of logic programs (Gelfond & Lifschitz 1988, Lifschitz et al. 2001, Turner 2003) in Section 2.1, hybrid knowledge bases (de Bruijn, Pearce, Polleres & Valverde 2010) in Section 2.2, and normal DL logic programs (Shen & Wang 2011) in Section 2.3. We then review the foundations of belief change in Section 2.4.

2.1 Logic Programming

Let $\mathcal{A}$ be a finite vocabulary of propositional atoms. A *rule* r over $\mathcal{A}$ has the form

$$a_1; \ldots; a_k; not\, b_1; \ldots; not\, b_l \leftarrow c_1, \ldots, c_m, not\, d_1, \ldots, not\, d_n. \tag{2.1}$$

Here, all $a_i, b_i, c_i, d_i \in \mathcal{A}$ and $k, l, m, n \geq 0$. The operators '*not*', ';', and ',' stand for default negation, disjunction, and conjunction, respectively. For convenience, let $H^+(r) = \{a_1, \ldots, a_k\}$, $H^-(r) = \{b_1, \ldots, b_l\}$, $B^+(r) = \{c_1, \ldots, c_m\}$, and $B^-(r) = \{d_1, \ldots, d_n\}$. If $k = 1$ and $l = m = n = 0$,

then r is called a *fact* and we omit '$\leftarrow$'; if $k = l = 0$, then r is a *constraint* and we denote the empty disjunction by $\bot$. Let $At(r)$ and $At(R)$ denote the set of all atoms that occur in a rule of the form (2.1) and in a set of rules R, respectively. A *(generalised) logic program* is a finite set of rules of the form (2.1). We write $\mathcal{LP}_{\mathcal{A}}$ for the class of all logic programs that can be constructed from $\mathcal{A}$.

An *interpretation* $Y \subseteq \mathcal{A}$ satisfies a program P, denoted by $Y \models P$, iff it is a model of all rules under the standard definition for propositional logic such that each rule represents a standard conditional and default negation is transcribed to classical negation. Let $Mod(P) = \{ Y \mid Y \models P \}$. An *answer set* (Gelfond & Lifschitz 1988) of a program P is any subset-minimal interpretation Y that satisfies the *reduct* P^Y of P with respect to Y, defined as:

$$P^Y = \{ H^+(r) \leftarrow B^+(r) \mid r \in P, H^-(r) \subseteq Y, \text{ and } B^-(r) \cap Y = \emptyset \}.$$

The set of all answer sets of P is denoted by $AS(P)$.

An *SE interpretation* is a tuple (X, Y) of interpretations with $X \subseteq Y \subseteq \mathcal{A}$. We usually write, e.g., (ab, ab) instead of $(\{a, b\}, \{a, b\})$ for legibility. Let $\mathcal{SE}$ be the set of all SE interpretations over $\mathcal{A}$. For any set S of SE interpretations, by $\overline{S}$ we denote the complement of S with respect to $\mathcal{SE}$, that is, $\overline{S} = \mathcal{SE} \setminus S$. An SE interpretation (X, Y) is an *SE model* (Turner 2003) of a program P iff $Y \models P$ and $X \models P^Y$. The set of all SE models of P is denoted by $SE(P)$ and P is *satisfiable* iff $SE(P) \neq \emptyset$. Often we drop explicit set notation for rules and their union, e.g., for rules $r, r' \in P$, we use $SE(r)$ to denote $SE(\{r\})$ and write $SE(r \cup r')$ instead of $SE(\{r\} \cup \{r'\})$. Note that $SE(P) = \bigcap_{r \in P} SE(r)$. Given two programs P and Q, we say that P is *strongly equivalent* (Lifschitz et al. 2001) to Q, denoted by $P \equiv_s Q$, iff $SE(P) = SE(Q)$, and P *implies* Q, denoted by $P \models_s Q$, iff $SE(P) \subseteq SE(Q)$. In the particular case of $SE(P) \subset SE(Q)$, we say that P *strictly implies* Q. The relation $\models_s$ is antitonic with respect to the program subset relation, i.e., $Q \subseteq P$ implies $P \models_s Q$. Furthermore, we write $\models_s P$ to express $SE(P) = \mathcal{SE}$.

SE models are a refinement of answer sets as they provide more information about the atoms in a program and their dependencies. For example, each of the following programs $P_1, P_2, \ldots, P_9$ has $\{\emptyset\}$ as the only answer set but the

sets of SE models are different for each program:

$P_1 = \{ \bot \leftarrow a. \}$ $SE(P_1) = \{(\emptyset, \emptyset), (\emptyset, b), (b, b)\}$

$P_2 = \{ \bot \leftarrow b. \}$ $SE(P_2) = \{(\emptyset, \emptyset), (\emptyset, a), (a, a)\}$

$P_3 = \{ a \leftarrow b. \}$ $SE(P_3) = \{(\emptyset, \emptyset), (\emptyset, a), (a, a), (\emptyset, ab), (a, ab), (ab, ab)\}$

$P_4 = \{ b \leftarrow a. \}$ $SE(P_4) = \{(\emptyset, \emptyset), (\emptyset, b), (b, b), (\emptyset, ab), (b, ab), (ab, ab)\}$

$P_5 = \{ \bot \leftarrow a, b. \}$ $SE(P_5) = \{(\emptyset, \emptyset), (\emptyset, a), (a, a), (\emptyset, b), (b, b)\}$

$P_6 = \{ \bot \leftarrow not\, a, b. \}$ $SE(P_6) = \{(\emptyset, \emptyset), (\emptyset, a), (a, a), (\emptyset, ab), (a, ab), (b, ab), (ab, ab)\}$

$P_7 = \{ \bot \leftarrow a, not\, b. \}$ $SE(P_7) = \{(\emptyset, \emptyset), (\emptyset, b), (b, b), (\emptyset, ab), (a, ab), (b, ab), (ab, ab)\}$

$P_8 = \{ a; not\, b. \}$ $SE(P_8) = \{(\emptyset, \emptyset), (\emptyset, a), (a, a), (a, ab), (ab, ab)\}$

$P_9 = \{ not\, a; b. \}$ $SE(P_9) = \{(\emptyset, \emptyset), (\emptyset, b), (b, b), (b, ab), (ab, ab)\}$

Informally, we can interpret the content of an SE model (X, Y) on a three-valued scale. Any atoms in X are true, any atoms not in Y are false, and any atoms in Y but not in X are undefined.

2.2 Hybrid Knowledge Bases

Hybrid knowledge bases can be viewed as extensions of logic programs. In general, a hybrid knowledge base is any combination of a nonmonotonic logic program with a monotonic theory. Usually, the theory component contains general information about a domain and the program component covers areas of a domain for which only incomplete information exists. Combinations of this form give rise to expressive formalisms and allow monotonic reasoning over the theory component to be integrated with nonmonotonic reasoning, which more closely resembles the natural process of reasoning performed by humans, over the program component. We introduce the formal definitions now.

Let $\mathcal{L} = (\mathcal{C}, \mathcal{P}_T \cup \mathcal{P}_P)$ be a function-free, first-order language, in which $\mathcal{C}$ is a set of constant symbols and $\mathcal{P}_T, \mathcal{P}_P$ are sets of predicate symbols such that $\mathcal{C}, \mathcal{P}_T, \mathcal{P}_P$ are pairwise disjoint. Let $\mathcal{L}$ further include the symbols '$\wedge$', '$\vee$', '$\rightarrow$', '$\neg$', '$\forall$', '$\exists$', the predicate symbol '$\approx$' for equality, a countably infinite set of variables, and the standard punctuation marks, as well as the symbols ';', ',', '*not*', '$\leftarrow$', and '$\bot$'. Atoms, formulas, closed formulas, and theories are defined in the usual way. We lift the syntax of rules to the first-order form as follows.

A *(first-order) rule* r over $\mathcal{L}$ has the form

$$A_1; \ldots ; A_k; not\, B_1; \ldots ; not\, B_l \leftarrow C_1, \ldots, C_m, not\, D_1, \ldots, not\, D_n. \qquad (2.2)$$

Here, all A_i, B_i, C_i, D_i are atoms from $\mathcal{L}$ and $k, l, m, n \geq 0$. We carry over the notations '$\bot$' if $k = l = 0$, $H^+(r) = \{A_1, \ldots, A_k\}$, $H^-(r) = \{B_1, \ldots, B_l\}$, $B^+(r) = \{C_1, \ldots, C_m\}$, and $B^-(r) = \{D_1, \ldots, D_n\}$. A *hybrid knowledge base* $K = (T, P)$ over $\mathcal{L}$ consists of a classical first-order theory T over the language subset $\mathcal{L}_T = (\mathcal{C}, \mathcal{P}_T)$, and a set of rules of the form (2.2) over $\mathcal{L}$ that constitute a *(first-order) program* P. We write $\mathcal{K}_\mathcal{L}$ for the class of all hybrid knowledge bases over $\mathcal{L}$.

An $\mathcal{L}$ *structure* is a pair $\mathcal{I} = (U, I)$, where the universe $U = (\mathcal{D}, \sigma)$ consists of a non-empty domain $\mathcal{D}$ and a function σ from $\mathcal{C} \cup \mathcal{D}$ to $\mathcal{D}$ such that $\sigma(d) = d$ for all $d \in \mathcal{D}$. For a tuple $t = (d_1, \ldots, d_n)$, we define $\sigma(t) = (\sigma(d_1), \ldots, \sigma(d_n))$. An interpretation I is a set of variable-free atoms that can be constructed from $(\mathcal{C} \cup \mathcal{D}, \mathcal{P}_T \cup \mathcal{P}_P)$. By $\mathcal{I}|_{\mathcal{L}'}$ we denote the restriction of $\mathcal{I}$ to a language $\mathcal{L}' \subseteq \mathcal{L}$. To obtain a ground instance of a rule in P, we replace each constant symbol c in the rule by $\sigma(c)$ and each variable in the rule by an element from $\mathcal{D}$. The set of all ground instances of rules in P is written $ground(P)$.

A $\mathbf{QHT}^s_=$ *(quantified here-and-there logic with static domains and equality) structure with respect to* $\mathcal{L}$ (Pearce & Valverde 2008) (or simply $\mathbf{QHT}^s_=$ structure if $\mathcal{L}$ is clear from the context) is a triple $\mathcal{M} = (U, H, T)$ such that (U, H) and (U, T) are $\mathcal{L}$ structures with $H \subseteq T$. Let $\mathcal{QHT}$ be the set of all $\mathbf{QHT}^s_=$ structures over $\mathcal{L}$. For any set S of $\mathbf{QHT}^s_=$ structures, by $\overline{S}$ we denote the complement of S with respect to $\mathcal{QHT}$, that is, $\overline{S} = \mathcal{QHT} \setminus S$. We define the satisfaction relation for a $\mathbf{QHT}^s_=$ structure $\mathcal{M}$ recursively as follows. Let $p(t_1, \ldots, t_n)$ be an atom and ϕ, ψ closed formulas constructed from $(\mathcal{C} \cup \mathcal{D}, \mathcal{P}_T \cup \mathcal{P}_P)$, x be a variable, and $w \in \{H, T\}$. Then

$\mathcal{M} \models_w p(t_1, \ldots, t_n)$ iff $p(\sigma(t_1), \ldots, \sigma(t_n)) \in w$,
$\mathcal{M} \models_w \phi \wedge \psi$ iff $\mathcal{M} \models_w \phi$ and $\mathcal{M} \models_w \psi$,
$\mathcal{M} \models_w \phi \vee \psi$ iff $\mathcal{M} \models_w \phi$ or $\mathcal{M} \models_w \psi$,
$\mathcal{M} \models_T \phi \rightarrow \psi$ iff $\mathcal{M} \not\models_T \phi$ or $\mathcal{M} \models_T \psi$,
$\mathcal{M} \models_H \phi \rightarrow \psi$ iff (i) $\mathcal{M} \models_T \phi \rightarrow \psi$ and (ii) $\mathcal{M} \not\models_H \phi$ or $\mathcal{M} \models_H \psi$,
$\mathcal{M} \models_w \neg\phi$ iff $\mathcal{M} \not\models_T \phi$,
$\mathcal{M} \models_T \forall x \phi(x)$ iff $\mathcal{M} \models_T \phi(d)$ for all $d \in \mathcal{D}$,

$\mathcal{M} \models_H \forall x\phi(x)$ iff $\mathcal{M} \models_T \forall x\phi(x)$ and $\mathcal{M} \models_H \phi(d)$ for all $d \in \mathcal{D}$,
$\mathcal{M} \models_w \exists x\phi(x)$ iff $\mathcal{M} \models_w \phi(d)$ for some $d \in \mathcal{D}$,
$\mathcal{M} \models_w t_1 \approx t_2$ iff $\sigma(t_1) = \sigma(t_2)$.

A $\mathbf{QHT}^s_=$ structure $\mathcal{M}$ is a $\mathbf{QHT}^s_=$*model* of a closed formula ϕ, denoted $\mathcal{M} \models \phi$, iff $\mathcal{M} \models_w \phi$ for each $w \in \{H, T\}$. A $\mathbf{QHT}^s_=$ structure $\mathcal{M}$ is a:

- $\mathbf{QHT}^s_=$ model of a theory T iff it is a model of all formulas in T;
- $\mathbf{QHT}^s_=$ model of a program P iff it is a $\mathbf{QHT}^s_=$ model of all rules in $ground(P)$ where each rule of the form (2.2) is interpreted as

$$C_1 \wedge \cdots \wedge C_m \wedge \neg D_1 \wedge \cdots \wedge \neg D_n \rightarrow A_1 \vee \cdots \vee A_k \vee \neg B_1 \vee \cdots \vee \neg B_l\,;$$

- $\mathbf{QHT}^s_=$ model of a hybrid knowledge base $K = (T, P)$ iff it is a $\mathbf{QHT}^s_=$ model of T and P.

The set of all $\mathbf{QHT}^s_=$ models of K is denoted by $QHT(K)$ and K is *satisfiable* iff $QHT(K) \neq \emptyset$. Given two hybrid knowledge bases K, K', we say that K is *equivalent* to K', denoted by $K \equiv K'$, iff $QHT(K) = QHT(K')$, and K *implies* K', denoted by $K \models K'$, iff $QHT(K) \subseteq QHT(K')$. We write $\models K$ to express $QHT(K) = \mathcal{QHT}$.

A hybrid knowledge base can be used to combine specific constraints that hold in a domain, expressed by a program component P, with more general information, captured by a theory component T. For instance, a company in which manufacturing is performed by humans and robots may have the components of Example 2.1 (modified from (de Bruijn et al. 2010)) in its hybrid knowledge base.

Example 2.1. Let $\mathcal{C} = \{JimBlock\}$, $\mathcal{P}_T = \{Worker, Person, IsMammal\}$, $\mathcal{P}_P = \{Machine\}$, and $K = (T, P)$ be a hybrid knowledge base over $(\mathcal{C}, \mathcal{P}_T \cup \mathcal{P}_P)$ such that

$$\begin{aligned} T &= \{\, \forall X Person(X) \rightarrow \forall X IsMammal(X)\,\}, \\ P &= \{\, Person(X) \leftarrow Worker(X),\, not\, Machine(X)., \\ &\qquad Worker(JimBlock).\,\}. \end{aligned}$$

We obtain $QHT(K) = \{\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3, \mathcal{M}_4, \mathcal{M}_5, \mathcal{M}_6, \mathcal{M}_7\}$ with

$$
\begin{aligned}
\mathcal{M}_1 = ((&\{JimBlock\}, \sigma),\\
&\{Worker(JimBlock), Person(JimBlock), IsMammal(JimBlock)\},\\
&\{Worker(JimBlock), Person(JimBlock), IsMammal(JimBlock)\}),\\
\mathcal{M}_2 = ((&\{JimBlock\}, \sigma),\\
&\{Worker(JimBlock), Person(JimBlock), IsMammal(JimBlock)\},\\
&\{Worker(JimBlock), Machine(JimBlock),\\
&Person(JimBlock), IsMammal(JimBlock)\}),\\
\mathcal{M}_3 = ((&\{JimBlock\}, \sigma),\\
&\{Worker(JimBlock), Machine(JimBlock)\},\\
&\{Worker(JimBlock), Machine(JimBlock)\}),\\
\mathcal{M}_4 = ((&\{JimBlock\}, \sigma),\\
&\{Worker(JimBlock), Machine(JimBlock)\},\\
&\{Worker(JimBlock), Machine(JimBlock),\\
&IsMammal(JimBlock)\}),\\
\mathcal{M}_5 = ((&\{JimBlock\}, \sigma),\\
&\{Worker(JimBlock), Machine(JimBlock)\},\\
&\{Worker(JimBlock), Machine(JimBlock),\\
&Person(JimBlock), IsMammal(JimBlock)\}),\\
\mathcal{M}_6 = ((&\{JimBlock\}, \sigma),\\
&\{Worker(JimBlock), Machine(JimBlock),\\
&IsMammal(JimBlock)\},\\
&\{Worker(JimBlock), Machine(JimBlock),\\
&Person(JimBlock), IsMammal(JimBlock)\}),\\
\mathcal{M}_7 = ((&\{JimBlock\}, \sigma),\\
&\{Worker(JimBlock), Machine(JimBlock),\\
&Person(JimBlock), IsMammal(JimBlock)\},\\
&\{Worker(JimBlock), Machine(JimBlock),\\
&Person(JimBlock), IsMammal(JimBlock)\}).
\end{aligned}
$$

■

2.3 Normal DL Logic Programs

We give only a brief overview of description logics (DLs) here and refer the reader to (Baader, Calvanese, McGuinness, Nardi & Patel-Schneider 2007) for details. The results in this work are applicable to a wide range of DL languages and thus, for simplicity, we consider *DL-Lite*$^{\mathcal{N}}_{bool}$ from the family of DL-Lite (Artale, Calvanese, Kontchakov & Zakharyaschev 2009).

Let $\Sigma = (\mathcal{C}, \Sigma_C \cup \Sigma_R)$ be a finite vocabulary of pairwise disjoint *individuals* in $\mathcal{C}$, which are names of singular elements in some domain of reference, *atomic concepts* in Σ_C, which are classes of individuals, and *atomic roles* in Σ_R, which represent relationships between two individuals, respectively (Rudolph 2011). *Complex roles* R and *complex concepts* C are formed according to the following syntax:

$$R \longleftarrow S \mid S^{-}\,,$$
$$C \longleftarrow \top \mid \bot \mid A \mid \neg C \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid \geqslant nR \mid \leqslant nR\,,$$

where $S \in \Sigma_R$, $A \in \Sigma_C$, and n is a positive integer. A *DL-Lite*$^{\mathcal{N}}_{bool}$ *knowledge base* L over Σ consists of a finite number of *inclusion axioms* of the form $C_1 \sqsubseteq C_2$ or $R_1 \sqsubseteq R_2$, called *TBox* of L, and *assertions* of the form $C(a)$ or $R(a, b)$ with $a, b \in \mathcal{C}$, called *ABox* of L.

The semantics of DL languages can be defined via first-order semantics by considering atomic concepts and roles as unary and binary predicates, respectively, individuals as constants, and complex concepts as first-order formulas. Details on this translation can be found in (Baader & Nutt 2007). Thus, we refer to atomic concepts and roles as *predicate symbols*, call L *consistent* if L has a first-order model, and write $L \models \alpha$, for any axiom α, if all models of L are models of α.

A *DL axiom* D is of the following form, where t, t_1, t_2 are each a variable or constant:

$$D = \begin{cases} C(t) \text{ or } \neg C(t)\,, \\ R(t_1, t_2) \text{ or } \neg R(t_1, t_2)\,, \\ C \sqsubseteq D. \end{cases}$$

If D contains no variables, then D is *ground*. If C is an atomic concept and R an atomic role, then $C(t)$ and $R(t_1, t_2)$ are called *atomic DL axioms*.

A *normal DL logic program* Π *relative to* $\mathcal{L}$ is a hybrid knowledge base $\Pi = (L, P)$ over $\mathcal{L} = (\mathcal{C}, \mathcal{P}_L \cup \mathcal{P}_P)$, where L is a DL knowledge base over the language subset $\mathcal{L}_L = (\mathcal{C}, \mathcal{P}_L)$, and P is a first-order program over the language subset $\mathcal{L}_P = (\mathcal{C}, \mathcal{P}_P \cup \Omega)$ with $\Omega \subseteq \mathcal{P}_L$, consisting of rules of the form (2.2) with $k = 1, l = 0$, and $m, n \geq 0$, such that A_1 is an atom (possibly an atomic DL axiom) and every C_i, D_i is an atom or a DL axiom.

To obtain a *ground instance* of a rule in Π, we replace each variable in the rule by a constant from $\mathcal{C}$. The set of all ground instances of rules in Π is written $ground(\Pi)$. The *Herbrand base* of Π relative to L, denoted HB_Π, is the set of all atoms that can be formed from predicate symbols in $\mathcal{P}_P \cup \Omega$ and elements of $\mathcal{C}$. The shared vocabulary Ω allows HB_Π to contain atoms where the predicate symbols are also in $\mathcal{P}_L$. Any atom in HB_Π that is not an atomic DL axiom is called a *pure atom*. Any subset of HB_Π is called an *interpretation of* Π *relative to* L, and for any such subset $\mathcal{S}$, we use $\overline{\mathcal{S}}$ to denote $HB_\Pi \setminus \mathcal{S}$, the complement of $\mathcal{S}$ in HB_Π. Whenever it is clear from the context, we will omit the specification "relative to L".

Given an atom $A \in HB_\Pi$, by $pred(A)$ we denote the predicate symbol of A. For any interpretation $\mathcal{I}$ of Π, let $\mathcal{I}|_\Omega = \{ A \mid A \in \mathcal{I} \text{ and } pred(A) \in \Omega \}$ and $\neg\mathcal{I}|_\Omega = \{ \neg A \mid A \in \mathcal{I} \text{ and } pred(A) \in \Omega \}$. We call $\mathcal{I}$ *consistent with* L iff $L \cup \mathcal{I}|_\Omega \cup \neg\overline{\mathcal{I}}|_\Omega$ is consistent. By a *literal* we mean a pure atom or ground DL axiom or its negation by default. The *satisfaction* of a literal l under $\mathcal{I}$ (relative to L), denoted $\mathcal{I} \models_L l$, is defined as follows:

- A is a pure atom from HB_Π
  - $\mathcal{I} \models_L A$ iff $A \in \mathcal{I}$;
  - $\mathcal{I} \models_L not\ A$ iff $\mathcal{I} \not\models_L A$.
- A is a ground DL axiom
  - $\mathcal{I} \models_L A$ iff $L \cup \mathcal{I}|_\Omega \cup \neg\overline{\mathcal{I}}|_\Omega \models A$;
  - $\mathcal{I} \models_L not\ A$ iff $\mathcal{I} \not\models_L A$.

For any rule r in $ground(\Pi)$, if $\mathcal{I} \models_L A$ for all $A \in B^+(r)$ and $\mathcal{I} \not\models_L A$ for all $A \in B^-(r)$, then $\mathcal{I}$ satisfies $body(r) = B^+(r) \cup \{ not\ A \mid A \in B^-(r) \}$. If $\mathcal{I}$ satisfies $head(r) = H^+(r)$ or does not satisfy $body(r)$, then $\mathcal{I}$ satisfies r. If $\mathcal{I}$ is consistent with L and satisfies all rules in $ground(\Pi)$, then $\mathcal{I}$ is a *model* of Π relative to L.

For any $\mathcal{E} \subseteq \mathcal{I}$, $\mathcal{E}$ *up to* $\mathcal{I}$ *satisfies* an atom or ground DL axiom A under L, written $(\mathcal{E},\mathcal{I}) \models_L A$, if for every $\mathcal{F}$ with $\mathcal{E} \subseteq \mathcal{F} \subseteq \mathcal{I}$, $\mathcal{F}$ satisfies A. If there does not exist an $\mathcal{F}$ with $\mathcal{E} \subseteq \mathcal{F} \subseteq \mathcal{I}$ such that $\mathcal{F}$ satisfies A, then $(\mathcal{E},\mathcal{I}) \models_L not\, A$. If $(\mathcal{E},\mathcal{I}) \models_L l$ holds for every literal l in $body(r)$, then $(\mathcal{E},\mathcal{I}) \models_L body(r)$.

Let

$$\mathcal{T}_\Pi(\mathcal{E},\mathcal{I}) = \{\, A \mid A \leftarrow body(r) \in ground(\Pi) \text{ and } (\mathcal{E},\mathcal{I}) \models_L body(r) \,\}$$

be an immediate consequence operator with $\mathcal{T}_\Pi^0(\emptyset,\mathcal{I}) = \emptyset$ and $\mathcal{T}_\Pi^{i+1}(\emptyset,\mathcal{I}) = \mathcal{T}_\Pi(\mathcal{T}_\Pi^i(\emptyset,\mathcal{I}),\mathcal{I})$. Then the sequence $\bigcup_{i=0}^{\infty} \mathcal{T}_\Pi^i(\emptyset,\mathcal{I})$ converges to a fixpoint, denoted by $\mathcal{T}_\Pi^\alpha(\emptyset,\mathcal{I})$. An interpretation $\mathcal{I}$ is a *(two-valued) answer set* of Π relative to L if $\mathcal{I}$ is a model of Π and for every $A \in \mathcal{I}$, either $A \in \mathcal{T}_\Pi^\alpha(\emptyset,\mathcal{I})$ or $L \cup \mathcal{T}_\Pi^\alpha(\emptyset,\mathcal{I})|_\Omega \cup \neg\overline{\mathcal{I}}|_\Omega \models A$. A normal DL logic program Π is *inconsistent* with respect to a DL knowledge base L iff it does not have a two-valued answer set.

Example 2.2. Let $L = \{A \sqsubseteq \neg C\}$ and

$$\Pi = \{\, A(X) \leftarrow not\, B(X).,\ D(X) \leftarrow A(X),\ not\, C(X). \,\}$$

with $\mathcal{P}_P = \{B, D\}$, $\mathcal{C} = \{a\}$, and $\Omega = \{A, C\}$. The DL axioms in Π are $A(X)$ and $C(X)$. We have $HB_\Pi = \{A(a), B(a), C(a), D(a)\}$ and $ground(\Pi) = \{\, A(a) \leftarrow not\, B(a).,\ D(a) \leftarrow A(a),\ not\, C(a). \,\}$. Six models exist for Π: $\{A(a), D(a)\}$, $\{A(a), B(a), D(a)\}$, $\{B(a)\}$, $\{B(a), C(a)\}$, $\{B(a), D(a)\}$, and $\{B(a), C(a),\ D(a)\}$. Only the first is an answer set of Π relative to L. ■

2.4 Belief Change

The AGM framework (Alchourrón et al. 1985, Gärdenfors 1988) defines *expansion*, *revision*, and *contraction* as the change operations on a body of beliefs held by an agent, called a *belief state* henceforth, in light of some new information. In an expansion, new beliefs are incorporated into a belief state, regardless of any inconsistencies that may arise. A revision operation also incorporates new beliefs into a belief state, but in such a way that the resulting belief state is consistent. This is achieved by discarding some existing beliefs. During a contraction, some beliefs in a belief state are removed without adding new beliefs.

In the AGM framework, a belief state is modelled as a *belief set*, defined as a set of sentences from some logic-based language $\mathcal{L}$ that is closed under logical consequence, i.e., when all beliefs implied by a knowledge base are explicitly represented in the knowledge base. Let K be a belief set, ϕ a sentence, $K_\perp$ denote the inconsistent belief set, and $Cn(\cdot)$ stand for a logical consequence function. The expansion of K by ϕ, written $K \oplus \phi$, is defined as $K \oplus \phi = Cn(K \cup \{\phi\})$. The AGM framework provides a set of postulates that any rational revision operator should satisfy. The postulates are listed as follows, where $\circledast$ represents a revision operator.

($\circledast$1) $K \circledast \phi$ is a belief set

($\circledast$2) $\phi \in K \circledast \phi$

($\circledast$3) $K \circledast \phi \subseteq K \oplus \phi$

($\circledast$4) If $\neg\phi \notin K$, then $K \oplus \phi \subseteq K \circledast \phi$

($\circledast$5) $K \circledast \phi = K_\perp$ iff $\vdash \neg\phi$

($\circledast$6) If $\phi_1 \equiv \phi_2$, then $K \circledast \phi_1 = K \circledast \phi_2$

($\circledast$7) $K \circledast (\phi \wedge \psi) \subseteq (K \circledast \phi) \oplus \psi$

($\circledast$8) If $\neg\psi \notin K \circledast \phi$, then $(K \circledast \phi) \oplus \psi \subseteq K \circledast (\phi \wedge \psi)$

($\circledast$1) requires that the outcome of a revision is a belief set. ($\circledast$2) states that the revising sentence is contained in the revised belief set. ($\circledast$3) asserts that a belief set revised by a sentence is always a subset of the belief set expanded by that sentence. ($\circledast$3) and ($\circledast$4) together state that revision coincides with expansion in cases when the revising sentence is consistent with the initial belief set. ($\circledast$5) guarantees that a revision outcome is consistent, unless the revising sentence is logically impossible. ($\circledast$6) ensures that logically equivalent sentences lead to the same revision outcomes. ($\circledast$7) and ($\circledast$8) together enforce K to be minimally changed in a revision by both ϕ and ψ, such that the outcome is the same as the expansion of $K \circledast \phi$ by ψ, provided that ψ is consistent with $K \circledast \phi$.

In the concrete case that a belief state is represented as a finite set of propositional formulas, the following set of postulates is equivalent to the set ($\circledast$1)–($\circledast$8) (Katsuno & Mendelzon 1991). Let ϕ, ψ, μ be propositional formulas.

($\circledast$1KM) $\phi \circledast \psi$ implies ψ

($\circledast$2KM) If $\phi \wedge \psi$ is satisfiable, then $\phi \circledast \psi \equiv \phi \wedge \psi$

($\circledast$3KM) If ψ is satisfiable, then $\phi \circledast \psi$ is satisfiable

($\circledast$4KM) If $\phi_1 \equiv \phi_2$ and $\psi_1 \equiv \psi_2$, then $\phi_1 \circledast \psi_1 \equiv \phi_2 \circledast \psi_2$

($\circledast$5KM) $(\phi \circledast \psi) \wedge \mu$ implies $\phi \circledast (\psi \wedge \mu)$

($\circledast$6KM) If $(\phi \circledast \psi) \wedge \mu$ is satisfiable, then $\phi \circledast (\psi \wedge \mu)$ implies $(\phi \circledast \psi) \wedge \mu$

($\circledast$1KM) requires that the revising formula can be derived from the revision outcome. ($\circledast$2KM) specifies that revision corresponds to conjunction whenever the revising formula is consistent with the formula to be revised. ($\circledast$3KM) guarantees consistency of a revision outcome whenever the revising formula is consistent. ($\circledast$4KM) states that revising logically equivalent formulas by logically equivalent formulas leads to logically equivalent results. ($\circledast$5KM) and ($\circledast$6KM) together stipulate that the revision by a conjunction leads to the same outcome as revising by one conjunct and then forming the conjunction with the other conjunct, provided that the conjunction thus formed is satisfiable.

The AGM framework also provides a set of postulates that any rational contraction operator should satisfy. The postulates are given below, where $\ominus$ represents a contraction operator.

($\ominus$1) $K \ominus \phi$ is a belief set

($\ominus$2) $K \ominus \phi \subseteq K$

($\ominus$3) If $\phi \notin K$, then $K \ominus \phi = K$

($\ominus$4) If $\nvdash \phi$, then $\phi \notin K \ominus \phi$

($\ominus$5) $K \subseteq (K \ominus \phi) \oplus \phi$

($\ominus$6) If $\phi_1 \equiv \phi_2$, then $K \ominus \phi_1 = K \ominus \phi_2$

($\ominus$7) $K \ominus \phi \cap K \ominus \psi \subseteq K \ominus \phi \wedge \psi$

($\ominus$8) If $\phi \notin K \ominus \phi \wedge \psi$, then $K \ominus \phi \wedge \psi \subseteq K \ominus \phi$

($\ominus$1) requires that the outcome of a contraction is a belief set. ($\ominus$2) ensures that no new beliefs are introduced during a contraction. ($\ominus$3) stipulates that the belief set remains unchanged during a contraction operation whenever the sentence to be contracted is not contained in it. ($\ominus$4) states that a contracting sentence is not a logical consequence of the contracted belief set, unless the sentence is a tautology. ($\ominus$5) requires that the original belief set can be recovered by expanding a contracted belief set by the sentence that was contracted. ($\ominus$6) ensures that logically equivalent sentences lead to the same contraction outcomes. ($\ominus$7) guarantees that any beliefs retained in both $K \ominus \phi$ and $K \ominus \psi$ are also retained in $K \ominus (\phi \wedge \psi)$. ($\ominus$8) specifies that any beliefs retained in a contraction by $\phi \wedge \psi$ are also retained in a contraction by ϕ, whenever ϕ is not retained.

The appropriateness of the Recovery postulate ($\ominus$5) within this set of contraction postulates has been discussed intensively (Fuhrmann 1991, Hansson 1991, Makinson 1987, Nayak 1994, Niederée 1991). To replace the Recovery postulate in expressing that no beliefs should be retracted unduly during a contraction operation, alternative postulates were proposed. Hansson (1991) offered the following postulate:

($\ominus$5r) If $\psi \in K \setminus (K \ominus \phi)$, then there is a set K' such that $K \ominus \phi \subseteq K' \subset K$ and $\phi \notin Cn(K')$ but $\phi \in Cn(K' \cup \{\psi\})$.

The Relevance postulate ($\ominus$5r) states that a sentence ψ should only be removed during the contraction of a sentence ϕ from K if ψ is relevant for implying ϕ. In the presence of ($\ominus$1)–($\ominus$3), ($\ominus$5) is equivalent to ($\ominus$5r) in propositional logic (Hansson 1991). More recently, Fermé, Krevneris & Reis (2008) presented the following Disjunctive Elimination postulate ($\ominus$5de):

($\ominus$5de) If $\psi \in K \setminus (K \ominus \phi)$, then $K \ominus \phi \nvdash \phi \vee \psi$.

According to ($\ominus$5de), a sentence ψ should only be removed during the contraction of a sentence ϕ from K if the contraction result does not imply the disjuntion of ϕ and ψ. In the presence of ($\ominus$2)–($\ominus$3), ($\ominus$5r) is equivalent to ($\ominus$5de) in propositional logic (Fermé et al. 2008).

Among the classic constructions to implement belief change are *partial meet contraction* (Alchourrón et al. 1985) and *epistemic entrenchment contraction* (Gärdenfors & Makinson 1988), which we recapitulate here. A set K' is a *remainder set* of a set $K \subseteq \mathcal{L}$ with respect to a sentence ϕ iff

a) $K' \subseteq K$,

b) $K' \nvdash \phi$, and

c) for any K'' with $K' \subset K'' \subseteq K : K'' \vdash \phi$.

The set of all remainder sets of K with respect to ϕ is denoted by $K \perp \phi$. A *selection function* γ for K is a function such that (i) if $K \perp \phi \neq \emptyset$, then $\emptyset \neq \gamma(K \perp \phi) \subseteq K \perp \phi$ and (ii) $\gamma(K \perp \phi) = \{K\}$ otherwise. A *partial meet contraction operator* $\ominus_\gamma$ for K is defined as: $K \ominus_\gamma \phi = \bigcap \gamma(K \perp \phi)$. The following representation theorem shows that the set of postulates ($\ominus$1)–($\ominus$6) exactly characterises the class of partial meet contraction operators.

Theorem 2.3. (Alchourrón et al. 1985) For any belief set K, $\ominus_\gamma$ is a partial meet contraction operator for K iff $\ominus_\gamma$ satisfies ($\ominus$1)–($\ominus$6).

By placing further restrictions on the selection function, the representation theorem can be extended to the full set of postulates. A transitively relational selection function γ' for K is determined by a transitive relation $\trianglelefteq$ over 2^K such that $\gamma'(K \perp \phi) = \{ K' \in K \perp \phi \mid K'' \trianglelefteq K' \text{ for all } K'' \in K \perp \phi \}$. A partial meet contraction operator $\ominus_{\gamma'}$ determined by a transitively relational selection function γ' is called a *transitively relational partial meet contraction operator*.

Theorem 2.4. (Alchourrón et al. 1985) For any belief set K, $\ominus_{\gamma'}$ is a transitively relational partial meet contraction operator for K iff $\ominus_{\gamma'}$ satisfies ($\ominus$1)–($\ominus$8).

A corresponding *(transitively relational) partial meet revision operator* $\circledast_\gamma$ ($\circledast_{\gamma'}$) that satisfies ($\circledast$1)–($\circledast$6) (($\circledast$1)–($\circledast$8)) can be obtained from a (transitively relational) partial meet contraction operator via the *Levi identity*: $K \circledast \phi = (K \ominus \neg\phi) \oplus \phi$ (Gärdenfors 1981, Levi 1977). The inverse identity, which constructs a contraction operator from a revision operator, is due to Harper (1976): $K \ominus \phi = K \cap (K \circledast \neg\phi)$.

Given a belief set K over $\mathcal{L}$, an *epistemic entrenchment associated with* K is any binary relation $\leqslant$ over $\mathcal{L}$ that satisfies the following conditions.

(EE1) If $\phi \leqslant \psi$ and $\psi \leqslant \chi$, then $\phi \leqslant \chi$

(EE2) If $\phi \vdash \psi$, then $\phi \leqslant \psi$

(EE3) $\phi \leqslant \phi \wedge \psi$ or $\psi \leqslant \phi \wedge \psi$ for any ϕ, ψ

(EE4) If $K \nvdash K_{\perp}$, then $\phi \notin K$ iff $\phi \leqslant \psi$ for all ψ

(EE5) If $\psi \leqslant \phi$ for all ψ, then $\vdash \phi$

The strict relation $\phi < \psi$ is understood as $\phi \leqslant \psi$ and $\psi \not\leqslant \phi$. (EE1) ensures the relation is transitive. (EE2) states that logically weaker sentences are at least as entrenched as stronger ones and logically equivalent sentences are equally entrenched. (EE2) together with (EE3) require the conjunction of two sentences to be equally entrenched as the least entrenched of the two. (EE4) specifies that any sentences not in the belief set are least entrenched. (EE5) states that tautologies are most entrenched.

An *epistemic entrenchment contraction operator* $\ominus_{\leqslant}$ for K is defined as: $\psi \in K \ominus_{\leqslant} \phi$ iff $\psi \in K$ and either $\vdash \phi$ or $\phi < \phi \vee \psi$, and satisfies ($\ominus$1)–($\ominus$8).

Theorem 2.5. (Gärdenfors 1988) Let K be a belief set and $\ominus_{\leqslant}$ an entrenchment contraction operator for K. Then $\ominus_{\leqslant}$ satisfies ($\ominus$1)–($\ominus$8).

A revision operator $\circledast$ that satisfies ($\circledast$1)–($\circledast$8) can be obtained via the *Levi identity*.

While the AGM approach provides an effective framework to conduct belief change, the representation of belief states in the form of belief sets has some shortcomings (see (Hansson 1999) for a detailed discussion). From a practical perspective, main drawbacks of belief sets are that they are generally large objects, since all logical consequences of all beliefs are contained, and that it is impossible to distinguish between inconsistent belief sets, as inconsistent belief sets consist of the entire language. *Belief bases* (Fuhrmann 1991, Hansson 1989, Rott 1992) are an alternative representation of belief states. A belief base is a set of sentences from $\mathcal{L}$ that is not necessarily closed under logical consequence.

Hansson (1993) defined a *partial meet base contraction operator* $-_{\gamma}$ for a belief base B as $B -_{\gamma} \phi = \bigcap \gamma(B \perp \phi)$ and showed that the following set of postulates exactly characterises the class of partial meet base contraction operators.

($-$1) $B - \phi \subseteq B$

($-$2) If $\nvdash \phi$, then $\phi \notin Cn(B - \phi)$

($-$3) If $\psi \in B \setminus (B - \phi)$, then there is a set B' such that $B - \phi \subseteq B' \subset B$ and $\phi \notin Cn(B')$ but $\phi \in Cn(B' \cup \{\psi\})$

(−4) If it holds for all $B' \subseteq B$ that $\phi \in Cn(B')$ iff $\psi \in Cn(B')$, then $B - \phi = B - \psi$

Theorem 2.6. (Hansson 1993) For any belief base B, $-_\gamma$ is a partial meet base contraction operator for B iff $-_\gamma$ satisfies (−1)–(−4).

Note that (−1), (−2), and (−3) in the belief base setting correspond directly to (⊖2), (⊖4), and (⊖5r) in the AGM setting, respectively. (−4) states that if any parts of B which imply ϕ also imply ψ, then the same parts of B will be retained in a contraction by ϕ as in a contraction by ψ.

He also defined a corresponding *partial meet base revision operator* $\divideontimes_\gamma$ for a belief base B as $B \divideontimes_\gamma \phi = (B -_\gamma \neg\phi) \cup \{\phi\}$ and showed that the following set of postulates exactly characterises the class of partial meet base revision operators.

(⁎1) $\phi \in B \divideontimes \phi$

(⁎2) $B \divideontimes \phi \subseteq B \cup \{\phi\}$

(⁎3) If $\psi \in B \setminus (B \divideontimes \phi)$, then there is a set B' such that $B \divideontimes \phi \subseteq B' \subset B \cup \{\phi\}$ and $\neg\phi \notin Cn(B')$ but $\neg\phi \in Cn(B' \cup \{\psi\})$

(⁎4) If it holds for all $B' \subseteq B$ that $B' \cup \{\phi\}$ is consistent iff $B' \cup \{\psi\}$ is consistent, then $B \cap (B \divideontimes \phi) = B \cap (B \divideontimes \psi)$

(⁎5) If $\nvdash \neg\phi$, then $\neg\phi \notin Cn(B \divideontimes \phi)$

Theorem 2.7. (Hansson 1993) For any belief base B, $\divideontimes_\gamma$ is a partial meet base revision operator for B iff $\divideontimes_\gamma$ satisfies (⁎1)–(⁎5).

The pendants to (⁎1) and (⁎2) in the AGM framework are (⊛2) and (⊛3), respectively. (⁎3) requires ψ to only be removed from B if it would otherwise make the revision outcome inconsistent. (⁎4) mandates that if any parts of B which are consistent with ϕ are also consistent with ψ, then the same parts of B will be retained in a revision by ϕ as in a revision by ψ. (⁎5) is a weaker version of (⊛5).

Williams (1994) proposed further belief change operators for belief bases, which rely on an ordering over the sentences contained in a belief base, called ensconcement. An *ensconcement associated with a belief base* B is any total preorder $\preccurlyeq$ on B that satisfies the following conditions.

($\preccurlyeq$1) For all $\phi \in B : \{ \psi \in B \mid \phi \prec \psi \} \nvdash \phi$

($\preccurlyeq$2) For all $\phi, \psi \in B : \phi \preccurlyeq \psi$ iff $\vdash \psi$

Condition ($\preccurlyeq$1) states that sentences which are strictly more ensconced than a sentence ϕ do not entail ϕ. Condition ($\preccurlyeq$2) requires any tautologies in the belief base to be most ensconced. The *proper cut of B for ϕ* is $cut_{\prec}(\phi) = \{ \psi \in B \mid \{ \chi \in B \mid \psi \preccurlyeq \chi \} \nvdash \phi \}$. An *ensconcement contraction operator* $\ominus_{\preccurlyeq}$ for B is defined as: $\psi \in B \ominus_{\preccurlyeq} \phi$ iff $\psi \in B$ and either $\vdash \phi$ or $cut_{\prec}(\phi) \cup \{\neg\phi\} \vdash \psi$. An *ensconcement revision operator* $\circledast_{\preccurlyeq}$ for B is defined as: $\psi \in B \circledast_{\preccurlyeq} \phi$ iff (i) $\psi = \phi$ or (ii) $\psi \in B$ and either $\vdash \neg\phi$ or $cut_{\prec}(\neg\phi) \cup \{\phi\} \vdash \psi$.

An ensconcement contraction operator $\ominus_{\preccurlyeq}$ satisfies (-1), (-2), and

(-5) If $\phi \notin Cn(B)$, then $B - \phi = B$

(-6) If $\phi_1 \equiv \phi_2$, then $B - \phi_1 = B - \phi_2$

(-7) $B - \phi \wedge \psi = B - \phi$ or $B - \phi \wedge \psi = B - \psi$ or $B - \phi \wedge \psi = B - \phi \cap B - \psi$

(-8) If $\psi \in B \setminus (B - \phi)$, then $B - \phi \nvdash \phi \vee \psi$

Theorem 2.8. (Fermé et al. 2008) Let B be a belief base and $\ominus_{\preccurlyeq}$ an ensconcement contraction operator for B. Then $\ominus_{\preccurlyeq}$ satisfies (-1), (-2), and (-5)–(-8).[1]

Postulates (-5), (-6), and (-8) correspond directly to ($\ominus$3), ($\ominus$6), and ($\ominus$5de) in the AGM setting, respectively. (-7) states that a contraction by a conjuntion is the result of contracting by the first of the conjuncts, the result of contracting by the second of the conjuncts, or the common part of these two results. In the belief base framework, the relationship between (-3) and (-8) is different to the one between ($\ominus$5r) and ($\ominus$5de) in the AGM framework: (-3) implies (-8) but not vice versa (Fermé et al. 2008).

[1]Please note that the proof of the representation theorem (Theorem 14 in (Fermé et al. 2008)) contains an error, as acknowledged by the authors. The theorem only holds in the direction from operator to postulates as stated above.

3

Related Work

In this chapter, we present an overview of existing work that is most relevant to our study at hand. We begin with a review of the distance-based approach to logic program revision, which can be considered state-of-the-art in belief change for logic programs and serves as the fundament for the motivation of our work. This is followed by a review of screened semi-revision and an overview of further approaches related to belief revision in logic programs. Finally, we look at existing work in the area of belief change in hybrid knowledge bases.

3.1 Distance-Based Revision in Logic Programs

One of the key developments for adapting the AGM framework of belief change to logic programs came with the *distance-based approach* to logic program revision (Delgrande, Schaub, Tompits & Woltran 2013). It is built on the monotonic SE semantics for logic programs and understands a belief state as the set of SE models of a program. In that work, the formula-based revision postulates (⊛1KM)–(⊛6KM) are translated to logic programs as follows, where a revision operator $*$ is a function from $\mathcal{LP}_\mathcal{A} \times \mathcal{LP}_\mathcal{A}$ to $\mathcal{LP}_\mathcal{A}$ and the expansion of P by Q, denoted $P \dot{+} Q$, is understood as $P \dot{+} Q = R$ such that $R \in \mathcal{LP}_\mathcal{A}$

and $SE(R) = SE(P) \cap SE(Q)$.

(∗1m) $P * Q \models_s Q$

(∗2m) If $P \dot{+} Q$ is satisfiable, then $P * Q \equiv_s P \dot{+} Q$

(∗3m) If Q is satisfiable, then $P * Q$ is satisfiable

(∗4m) If $P_1 \equiv_s P_2$ and $Q_1 \equiv_s Q_2$, then $P_1 * Q_1 \equiv_s P_2 * Q_2$

(∗5m) $(P * Q) \dot{+} R \models_s P * (Q \dot{+} R)$

(∗6m) If $(P * Q) \dot{+} R$ is satisfiable, then $P * (Q \dot{+} R) \models_s (P * Q) \dot{+} R$

The approach adapts two revision operators from classic belief change to logic programs, namely, Dalal's revision operator (Dalal 1988) and Satoh's revision operator (Satoh 1988). Informally, to revise a program P by a program Q, the operators return those SE models from the set of SE models of Q that are closest to the SE models of P, where closeness is determined by Dalal's or Satoh's notion of distance. Delgrande, Schaub, Tompits & Woltran (2013) identified that the adaptation of Satoh's revision operator gives more intuitive results than the adaptation of Dalal's revision operator, so we will focus on the former here. This restriction has no effect on our later discussions.

We briefly restate the definition and main result of the distance-based approach. Let Δ stand for the symmetric difference between two sets X, Y, that is, $X \Delta Y = (X \setminus Y) \cup (Y \setminus X)$. For any two pairs of sets $(X, X'), (Y, Y')$, let

$$
\begin{aligned}
&(X, X')\Delta(Y, Y') = (X\Delta Y, X'\Delta Y');\\
&(X, X') \subseteq (Y, Y') \text{ iff } X' \subseteq Y', \text{ and if } X' = Y', \text{ then } X \subseteq Y;\\
&(X, X') \subset (Y, Y') \text{ iff } (X, X') \subseteq (Y, Y') \text{ and } (Y, Y') \nsubseteq (X, X').
\end{aligned}
$$

For any two sets E, E', let

$$
\begin{aligned}
\sigma(E, E') = \{\, A_1 \in E \mid \exists B_1 \in E' \text{ such that}\\
\forall A_2 \in E, \forall B_2 \in E' : A_1 \Delta B_1 \subseteq A_2 \Delta B_2 \,\}.
\end{aligned}
$$

Definition 3.1. (Delgrande, Schaub, Tompits & Woltran 2013) Let $P, Q \in \mathcal{LP}_{\mathcal{A}}$. The revision of P by Q, denoted $P \star Q$, is defined as $P \star Q = R$ such that $R \in \mathcal{LP}_{\mathcal{A}}$ and $SE(R) = SE(Q)$ if $SE(P) = \emptyset$, and otherwise

$$SE(R) = \{\, (X, Y) \mid Y \in \sigma(Mod(Q), Mod(P)), X \subseteq Y,$$

and if $X \subset Y$, then $(X, Y) \in \sigma(SE(Q), SE(P))\,\}$.

Theorem 3.2. (Delgrande, Schaub, Tompits & Woltran 2013) The revision operator $\star$ satisfies ($*$1m)–($*$5m).

The distance-based approach was extended by two representation theorems (Delgrande, Peppas & Woltran 2013, Schwind & Inoue 2013), stating that any logic program revision operator satisfying ($*$1m)–($*$6m) plus some additional conditions can be suitably characterised by some preorder over a set of SE models.

3.2 Screened Semi-Revision in Logic Programs

The *screened semi-revision approach* for logic programs (Krümpelmann & Kern-Isberner 2012) is based on answer set semantics and aligns itself with the belief base framework. The approach assumes a belief state to be the set of rules belonging to a program and combines adaptations of the constructions of semi-revision (Hansson 1997) and screened revision (Makinson 1997) into a screened consolidation operation for logic programs. The consolidation operator first finds all maximal subsets of one program that are consistent with a second program under answer set semantics, then selects exactly one of these subsets, and returns this subset together with the second program as the outcome.

We review the formal definitions of the screened consolidation operator and the main result here. Let $P \in \mathcal{LP}_{\mathcal{A}}$ and $Q \subseteq P$. The set of screened remainder sets of P with respect to Q is

$$P\bot_! Q = \{\, R \mid Q \subseteq R \subseteq P, AS(R) \neq \emptyset \text{ and for all } R' \text{ with } R \subset R' \subseteq P : AS(R') = \emptyset \,\}.$$

A maxichoice selection function γ_P for P is a function such that for any $Q \in \mathcal{LP}_{\mathcal{A}}$: (i) if $P\bot_! Q \neq \emptyset$, then $\gamma_P(P\bot_! Q) = R$ for some $R \in P\bot_! Q$, and (ii) if $P\bot_! Q = \emptyset$, then $\gamma_P(P\bot_! Q) = P$.

Definition 3.3. (Krümpelmann & Kern-Isberner 2012) Let $P, Q \in \mathcal{LP}_{\mathcal{A}}$ and γ_P be a maxichoice selection function for P. A screened consolidation operator $!_{\gamma_P}$ for P is defined as $P!_{\gamma_P} Q = \gamma_P(P\bot_! Q)$.

The authors propose the following adaptation of partial meet base revision postulates that any screened consolidation operator ! should satisfy, where ! is a function from $\mathcal{LP}_\mathcal{A} \times \mathcal{LP}_\mathcal{A}$ to $\mathcal{LP}_\mathcal{A}$, and show that $!_{\gamma_P}$ is exactly characterised by these postulates.

(!1) $Q \subseteq P\,!\,Q$

(!2) $P\,!\,Q \subseteq P$

(!3) If $r \in P \setminus (P\,!\,Q)$, then $AS(P\,!\,Q) \neq \emptyset$ and $AS(P\,!\,Q \cup \{r\}) = \emptyset$

(!4) If it holds for all $P' \subseteq P$ that $AS(P' \cup Q) \neq \emptyset$ iff $AS(P' \cup R) \neq \emptyset$, then $P \cap ((P \cup Q)\,!\,Q) = P \cap ((P \cup R)\,!\,R)$

(!5) If there exists some P' such that $Q \subseteq P' \subseteq P$ and $AS(P') \neq \emptyset$, then $AS(P\,!\,Q) \neq \emptyset$

Theorem 3.4. (Krümpelmann & Kern-Isberner 2012) For any $P \in \mathcal{LP}_\mathcal{A}$, $!_{\gamma_P}$ is a screened consolidation operator for P iff $!_{\gamma_P}$ satisfies (!1)–(!5).

3.3 Other Approaches to Belief Change in Logic Programs

Besides the distance-based revision and screened semi-revision approaches, few other methods for logic program revision have been proposed. The *program-level approach* to logic program revision (Delgrande 2010) is also based on answer set semantics and assumes the beliefs that make up a belief state to be the answer sets of a program. The revision operation relies on extending the standard answer set semantics to three-valued answer set semantics for determining the outcome. To revise a program P by a program Q, for each three-valued answer set X of Q, all maximal subsets R of P are selected such that X is a subset of each three-valued answer set X' of $R \cup Q$. The revision operation returns a set of answer sets that correspond to each X' as the result. The author argues that the AGM revision postulates ($\circledast$3), ($\circledast$4), ($\circledast$7), and ($\circledast$8) are inappropriate in the context of nonmonotonic semantics. An adaptation of the remaining postulates is fulfilled by the revision operation.

A relative of belief revision is *belief update* (Katsuno & Mendelzon 1992). The difference between these two is usually understood in the way that belief

revision addresses changes to a belief state brought about by some new information about a static world, whereas belief update covers changes to a belief state due to dynamics in the world described by the belief state. Similar to belief revision, the belief update framework prescribes a set of postulates that each rational update operator should satisfy and provides a construction that complies with it. A number of update operators for logic programs have been proposed, which differ greatly in the degree of alignment to the classic belief update framework. On the whole, these approaches rely predominantly on syntactic transformations of programs and return a set of answer sets instead of an updated program as the outcome. The landscape of update operators has already been reviewed exhaustively in other places, for example, detailed overviews are given by Delgrande, Schaub, Tompits & Wang (2004) and Slota (2012). Of interest in the current context is the *exception-based update* approach (Slota & Leite 2012), which introduces *RE (robust equivalence) models* as an extension of SE models. An RE model of a program P is any SE interpretation (X, Y) such that $X \models P^Y$. The authors regard a belief state as the collection of the sets of RE models of the rules in a program. In the update operation, exceptions in the form of RE models are added to those sets of RE models of the initial program that are incompatible with the RE models of the updating program. Incompatibilities between two sets of RE models are determined by differences in the truth values of atoms occurring in both sets. The update operator satisfies the majority of the update postulates adapted to logic programs.

3.4 Approaches to Belief Change in Hybrid Knowledge Bases

A number of approaches to integrate rule-based languages with first-order theories, and description logic ontologies in particular, have been proposed to date (overviews can be found in Drabent, Eiter, Ianni, Krennwallner, Lukasiewicz & Małuszyński 2009, Hitzler & Parsia 2009, Krisnadhi, Maier & Hitzler 2011). The approaches differ in the way rules and theories are combined and how reasoning is performed. They can be categorised as tight, loose, or hybrid rule formalisms (Wang, You, Yuan, Shen & Zhang 2012). The first type combines rules and theories in a single nonmonotonic formalism where rule predicates and theory predicates are specified in the same language, e.g., as found in

disjunctive dl-programs (Lukasiewicz 2010) and embeddings of logic programs into autoepistemic logic (Bruijn, Eiter, Polleres & Tompits 2011). In a loose integration, a rule r of the rule component can include queries to the theory. The result of a query is then substituted for the query predicate in r, and r is evaluated under the semantics of the rule language. Examples are dl-programs (Eiter, Ianni, Lukasiewicz, Schindlauer & Tompits 2008, Eiter, Ianni, Lukasiewicz & Schindlauer 2011) and HEX-programs (Eiter, Ianni, Schindlauer & Tompits 2005). Hybrid rules fall between these two categories as they allow rule predicates to be evaluated under nonmonotonic semantics and theory predicates under classical semantics, e.g., as defined in $\mathcal{DL}$+log (Rosati 2006) and hybrid rules (Drabent & Małuszyński 2007).

Among the range of integration approaches, the most expressive ones include the two methods introduced in the previous chapter, hybrid knowledge bases under $\mathbf{QHT}^s_=$ semantics and normal DL logic programs under three-valued answer set semantics, as well as *hybrid MKNF knowledge bases* (Motik & Rosati 2010, Knorr, Alferes & Hitzler 2011). Hybrid MKNF knowledge bases are based on the logic of minimal knowledge and negation-as-failure (MKNF) (Lifschitz 1991) and provide a tight combination of description logics with first-order and nonmonotonic rules. The approach allows reasoning under both the open-world and closed-world assumptions, making nonmonotonic inferences in the presence of first-order predicates possible. This is achieved by extending first-order atoms with the modal operators **K** and **not**.

To the best of our knowledge, the only existing work regarding changes to a hybrid knowledge base is the *update semantics for hybrid MKNF knowledge bases* (Slota, Leite & Swift 2011). It is a modular combination of the stable model semantics for hybrid MKNF knowledge bases and existing semantics for rule updates and description logic updates. The update operation is performed in two steps. First, the knowledge base is split into a hierarchical sequence of layers, similar to the process of logic program stratification, such that all information required for the evaluation of axioms and rules in a certain layer is declared in the same layer or in some higher layer (with acyclicity of axioms and rules implicitly assumed). Additionally, the splitting operation has to fulfill the condition that each layer contains either:

- only rules but no DL axioms (the layer is said to be $\mathcal{G}$*-reducible*), or
- only DL axioms and, possibly in addition, negation-free rules whose body

predicate symbols can be found in a higher layer (the layer is called $\mathcal{O}$-*reducible*).

In the second step, the models for each layer are computed, starting at the top layer and proceeding downwards:

- for $\mathcal{G}$-*reducible* layers, the refined dynamic stable model semantics (Alferes, Banti, Brogi & Leite 2005) is applied, and
- for $\mathcal{O}$-*reducible* layers, the minimal change update semantics (Winslett 1990) is applied.

Both steps have to be executed when the hybrid MKNF knowledge base is initially created and then each time a component is updated. The update method can thus be considered a divide-and-conquer operation that assembles the models of the updated knowledge base incrementally by evaluating its components under either first-order semantics or the stable model semantics with the aid of existing rule and description logic update operators.

4

Adapting the Belief Change Frameworks

Before we set out to define new constructions of logic program belief change, we translate the ideas of the classic belief change frameworks to the logic program setting. We assume that a belief state is represented in the form of a program from $\mathcal{LP}_{\mathcal{A}}$ and new information to expand/revise/contract this program comes in the form of another program from $\mathcal{LP}_{\mathcal{A}}$. Even though a consequence relation for logic programs under SE semantics exists (Eiter, Fink, Tompits & Woltran 2004, Wong 2008), logic programs are per se not closed under logical consequence. Thus, it seems natural to align logic program belief change with the belief base framework. However, as the majority of previous approaches to logic program revision have focussed on adapting the AGM framework, we will consider it here as well to enable us to draw proper comparisons.

As pointed out above, in the case of propositional knowledge bases the set of formula-based revision postulates ($\circledast$1KM)–($\circledast$6KM) is equivalent to the set ($\circledast$1)–($\circledast$8). However, in the context of logic programs, it is worth having a closer look at the relationship between these two sets. Besides adap-

tations for the purpose of logic program 'update' operations under answer set semantics (Eiter, Fink, Sabbatini & Tompits 2002) or N_2 logic (Osorio & Cuevas 2007), we find that postulates for logic program revision have usually been built on the formula-based revision postulates ($\circledast$1KM)–($\circledast$6KM) until now (Delgrande, Schaub, Tompits & Woltran 2013, Delgrande, Peppas & Woltran 2013, Schwind & Inoue 2013). They are given in the form of ($*$1m)–($*$6m) (see Section 3.1). We will now present a new set of postulates for logic program revision operators, based on the original AGM postulates ($\circledast$1)–($\circledast$8), and then explain their relationship to ($*$1m)–($*$6m). We will discover that the equivalence of the two sets of postulates under propositional logic does not carry over to logic programs, that, in fact, the adaptation of ($\circledast$1)–($\circledast$8) to logic programs leads to postulates that are in most cases stricter than ($*$1m)–($*$6m), since $Q \subseteq P$ implies $P \models_s Q$ but the converse does not hold.

Let $P, Q, R \in \mathcal{LP}_\mathcal{A}$ and a revision operator $*$ be a function from $\mathcal{LP}_\mathcal{A} \times \mathcal{LP}_\mathcal{A}$ to $\mathcal{LP}_\mathcal{A}$. In the following, we understand expansion, denoted by the operator $+$, as $P + Q = P \cup Q$.

($*$1) $P * Q \in \mathcal{LP}_\mathcal{A}$

($*$2) $Q \subseteq P * Q$

($*$3) $P * Q \subseteq P + Q$

($*$4) If $P + Q$ is satisfiable, then $P + Q \subseteq P * Q$

($*$5) $P * Q$ is satisfiable iff Q is satisfiable

($*$6) If $Q \equiv_s R$, then $P * Q \equiv_s P * R$

($*$7) $P * (Q + R) \subseteq (P * Q) + R$

($*$8) If $(P * Q) + R$ is satisfiable, then $(P * Q) + R \subseteq P * (Q + R)$

($*$1) is a basic but nonetheless crucial condition that is included in previous adaptations by the requirement that $*$ is a function from $\mathcal{LP}_\mathcal{A} \times \mathcal{LP}_\mathcal{A}$ to $\mathcal{LP}_\mathcal{A}$. ($*$2) stipulates that the revising program is always included in a revision outcome and is a stronger version of ($*$1m) since ($*$2) implies ($*$1m) but not vice versa. ($*$3) requires that a revision outcome never contains elements not in P or Q. This condition is covered by previous adaptations only for the case that $P + Q$ is satisfiable in ($*$2m). Together, ($*$3) and ($*$4)

state that revision coincides with expansion if $P+Q$ is satisfiable and thus imply ($*$2m) but not vice versa. ($*$5) is the stricter, biconditional version of ($*$3m) emphasising the consistency of a revision outcome. ($*$6) guarantees that revision by strongly equivalent programs leads to strongly equivalent results. Unlike belief sets, logic programs are not closed under logical consequence, so that we cannot expect from two different bodies of information that are merely strongly equivalent to be equal after a revision. Thus, the consequent is translated as "$P * Q \equiv_s P * R$" from the original postulate ($\circledast$6). ($*$4m) is stricter than ($*$6). ($*$7) and ($*$8) capture the minimal change condition and imply ($*$5m) and ($*$6m) but not vice versa, respectively.

A translation of the AGM contraction postulates to logic programs is listed below, where $P, Q, R \in \mathcal{LP}_\mathcal{A}$ and a contraction operator $\dot{-}$ is a function from $\mathcal{LP}_\mathcal{A} \times \mathcal{LP}_\mathcal{A}$ to $\mathcal{LP}_\mathcal{A}$.

($\dot{-}$1) $P \dot{-} Q \in \mathcal{LP}_\mathcal{A}$

($\dot{-}$2) $P \dot{-} Q \subseteq P$

($\dot{-}$3) If $P \not\models_s Q$, then $P \dot{-} Q = P$

($\dot{-}$4) If $\not\models_s Q$, then $P \dot{-} Q \not\models_s Q$

($\dot{-}$5) $P \subseteq (P \dot{-} Q) + Q$

($\dot{-}$6) If $Q \equiv_s R$, then $P \dot{-} Q = P \dot{-} R$

($\dot{-}$7) $(P \dot{-} Q) \cap (P \dot{-} R) \subseteq P \dot{-} (Q + R)$

($\dot{-}$8) If $P \dot{-} (Q + R) \not\models_s Q$, then $P \dot{-} (Q + R) \subseteq P \dot{-} Q$

Any contraction outcome is required to be a logic program by ($\dot{-}$1) and a subset of the initial program by ($\dot{-}$2). By ($\dot{-}$3), if the beliefs to be contracted are not implied by the initial belief state, then nothing is to be retracted. ($\dot{-}$4) requires that the beliefs to be contracted are not implied by the contracted belief state, unless they are tautologies. ($\dot{-}$5) states that all parts of the initial program P that are discarded in a contraction by Q can be recovered by a subsequent expansion with Q. ($\dot{-}$6) ensures that contraction by strongly equivalent programs leads to the same outcomes. ($\dot{-}$7) demands that any parts retained in both $P \dot{-} Q$ and $P \dot{-} R$ are also retained in $P \dot{-} (Q + R)$. Whenever Q is not implied by the result of a contraction by $Q + R$, then ($\dot{-}$8) states that this result is also retained in a contraction by Q alone.

($\dot{-}1$), ($\dot{-}2$), ($\dot{-}5$), ($\dot{-}6$), and ($\dot{-}7$) are direct translations of ($\ominus 1$), ($\ominus 2$), ($\ominus 5$), ($\ominus 6$), and ($\ominus 7$), respectively. In the adaptation of ($\ominus 3$) to ($\dot{-}3$), we use "$P \not\models_s Q$" instead of "$Q \not\subseteq P$". Belief sets are closed under logical consequence, which means $K \vdash \phi$ iff $\phi \in K$, so that either can be used as the condition in ($\ominus 3$). However, for logic programs under SE semantics, we can conclude $P \models_s Q$ from $Q \subseteq P$ but not $Q \subseteq P$ from $P \models_s Q$. Therefore, it is more appropriate to use "$P \not\models_s Q$" than "$Q \not\subseteq P$", as $P \not\models_s Q$ implies $Q \not\subseteq P$. For the same reason is "$P \dot{-} Q \not\models_s Q$" used instead of "$Q \not\subseteq P \dot{-} Q$" in ($\dot{-}4$), and "$P \dot{-} (Q + R) \not\models_s Q$" instead of "$Q \not\subseteq P \dot{-} (Q + R)$" in ($\dot{-}8$).

We now adapt the belief base revision postulates ($\circledast 1$)–($\circledast 5$) to logic programs, where again $P, Q, R \in \mathcal{LP}_{\mathcal{A}}$ and a revision operator $*$ is a function from $\mathcal{LP}_{\mathcal{A}} \times \mathcal{LP}_{\mathcal{A}}$ to $\mathcal{LP}_{\mathcal{A}}$.

($*1$b) $Q \subseteq P * Q$

($*2$b) $P * Q \subseteq P + Q$

($*3$b) If $r \in P \setminus (P * Q)$, then there exists a program P' such that $P * Q \subseteq P' \subset P + Q$ and P' is satisfiable but $P' \cup \{r\}$ is not satisfiable

($*4$b) If it holds for all $P' \subseteq P$ that $P' + Q$ is satisfiable iff $P' + R$ is satisfiable, then $P \cap (P * Q) = P \cap (P * R)$

($*5$b) If Q is satisfiable, then $P * Q$ is satisfiable

All five postulates are direct translations of ($\circledast 1$)–($\circledast 5$). We have ($*1$b) = ($*2$) and ($*2$b) = ($*3$). ($*3$b) requires a rule r to be removed during a revision of P by Q if r contributes to making P irreconcilable with Q. ($*4$b) states that if all subsets of P that agree with Q also agree with R, then the same elements of P will be retained in a revision by Q as in a revision by R. ($*5$b) guarantees satisfiability of the revision result whenever the revising program itself is satisfiable. ($*4$b) is a stronger version of ($*6$) and ($*5$b) is a weaker version of ($*5$).

Finally, we translate the belief base contraction postulates (-1)–(-8) to logic programs, where $P, Q, R \in \mathcal{LP}_{\mathcal{A}}$ and a contraction operator $\dot{-}$ is a function from $\mathcal{LP}_{\mathcal{A}} \times \mathcal{LP}_{\mathcal{A}}$ to $\mathcal{LP}_{\mathcal{A}}$.

($\dot{-}1$b) $P \dot{-} Q \subseteq P$

($\dot{-}2$b) If $\not\models_s Q$, then $P \dot{-} Q \not\models_s Q$

(∸3b) If $r \in P \setminus (P \dot{-} Q)$, then there exists a program P' such that $P \dot{-} Q \subseteq P' \subset P$ and $P' \not\models_s Q$ but $P' \cup \{r\} \models_s Q$

(∸4b) If it holds for all $P' \subseteq P$ that $P' \not\models_s Q$ iff $P' \not\models_s R$, then $P \dot{-} Q = P \dot{-} R$

(∸5b) If $P \not\models_s Q$, then $P \dot{-} Q = P$

(∸6b) If $Q \equiv_s R$, then $P \dot{-} Q = P \dot{-} R$

(∸7b) $P \dot{-} (Q + R) = P \dot{-} Q$ or $P \dot{-} (Q + R) = P \dot{-} R$ or $P \dot{-} (Q + R) = (P \dot{-} Q) \cap (P \dot{-} R)$

(∸8b) If $r \in P \setminus (P \dot{-} Q)$, then $SE(P \dot{-} Q) \not\subseteq SE(Q) \cup SE(r)$

These postulates are direct translations of (−1)–(−8). We have (∸1b) = (∸2), (∸2b) = (∸4), (∸5b) = (∸3), and (∸6b) = (∸6). (∸3b) requires a rule r to only be removed during the contraction of P by Q if r somehow contributes to implying Q. (∸4b) states that if exactly those subsets of P that do not imply Q also do not imply R, then the same elements of P will be retained in a contraction by Q as in a contraction by R. As for revision, (∸4b) is again a stronger version of (∸6). (∸7b) specifies that a contraction by two programs is the outcome of contracting the first program, the outcome of contracting the second program, or the intersection of these two outcomes. (∸8b) stipulates that r should only be removed if the contracted program has at least one SE model that is not an SE model of Q or r.

While the postulates (⊖5r) and (⊖5de) are equivalent in the presence of certain other postulates in the AGM framework for propositional logic, the relationship between the postulates (∸3b) and (∸8b) for logic programs is hierarchical, similar as in the belief base framework. Satisfaction of (∸3b) implies satisfaction of (∸8b) but not vice versa.

Proposition 4.1. Let $\dot{-}$ be a contraction operator on $\mathcal{LP}_{\mathcal{A}}$. If $\dot{-}$ satisfies (∸3b), then it satisfies (∸8b).

The other direction does not hold as evidenced later by Example 6.18.

We adapted the AGM and belief base postulates to logic programs in such a way that the revising or contracting beliefs are entire programs, not just individual rules. We have thus essentially defined conditions for what is known in classic belief change as choice contraction (Fuhrmann & Hansson 1994) and revision. In a choice contraction, a set of sentences may be contracted by

either a single sentence or a set of sentences, for example, $K \ominus \{\phi, \psi\}$. Yet, since adapting the postulates that exactly characterise partial meet choice contractions would yield the postulates ($\dot{-}$1b)–($\dot{-}$3b) and ($\dot{-}$6b), we do not have to pursue this distinction here further.

The Levi identity (Gärdenfors 1981, Levi 1977) allows us to construct a revision from a contraction and the Harper identity (Harper 1976) a contraction from a revision. In their propositional form, they use the classical negation of a sentence α to construct a revision/contraction by α. For logic programs, we do not have the classical negation of a program at at our disposal. However, we can abstractly represent the Levi and Harper identities for logic programs as follows, where $\overline{Q} \in \mathcal{LP}_{\mathcal{A}}$ iff $Q \in \mathcal{LP}_{\mathcal{A}}$ and $SE(\overline{Q}) = \overline{SE(Q)}$.

Definition 4.2. Let $P, Q \in \mathcal{LP}_{\mathcal{A}}$. If $*$ is a revision operator for P and $\dot{-}$ a contraction operator for P, then

$$P * Q = (P \dot{-} \overline{Q}) + Q \quad \text{(Levi identity)}$$
$$P \dot{-} Q = P \cap (P * \overline{Q}) \quad \text{(Harper identity)}$$

We now extend our sets of postulates to hybrid knowledge bases. Since hybrid knowledge bases are also not closed under logical consequence and no previous approaches to belief change in hybrid knowledge bases under the AGM framework exist, we only concentrate on the belief base framework. We first need to extend the expansion operator to hybrid knowledge bases. Let $+^h$ be an expansion operator for a hybrid knowledge base $K = (T, P)$ such that for any hybrid knowledge base $K' = (T', P')$, $K +^h K' = (T \cup T', P \cup P')$. In addition, we need to define the subset relation and the intersection and set difference operations for hybrid knowledge bases. For any two pairs of sets $K = (T, P)$ and $K' = (T', P')$, let $K \subseteq K'$ iff $T \subseteq T'$ and $P \subseteq P'$, $K \subset K'$ iff $K \subseteq K'$ and $K' \nsubseteq K$, $K \cap K' = (T \cap T', P \cap P')$, and $K \setminus K' = (T \setminus T', P \setminus P')$. We can now extend the belief base revision postulates ($*$1b)–($*$5b) and contraction postulates ($\dot{-}$1b)–($\dot{-}$8b) to hybrid knowledge bases as follows. Let $K, K', K'' \in \mathcal{K}_{\mathcal{L}}$ and a revision operator $*^h$ and a contraction operator $\dot{-}^h$ be functions from $\mathcal{K}_{\mathcal{L}} \times \mathcal{K}_{\mathcal{L}}$ to $\mathcal{K}_{\mathcal{L}}$.

($*^h$1b) $K' \subseteq K *^h K'$

($*^h$2b) $K *^h K' \subseteq K +^h K'$

($*^h$3b) If $\kappa \subseteq K \setminus (K *^h K')$, then there exists a hybrid knowledge base K_1 such that $K *^h K' \subseteq K_1 \subset K +^h K'$ and K_1 is satisfiable but $K_1 \cup \kappa$ is not satisfiable

($*^h$4b) If it holds for all $\kappa \subseteq K$ that $\kappa +^h K'$ is satisfiable iff $\kappa +^h K''$ is satisfiable, then $K \cap (K *^h K') = K \cap (K *^h K'')$

($*^h$5b) If K' is satisfiable, then $K *^h K'$ is satisfiable

($\dot{-}^h$1b) $K \dot{-}^h K' \subseteq K$

($\dot{-}^h$2b) If $\not\models K'$, then $K \dot{-}^h K' \not\models K'$

($\dot{-}^h$3b) If $\kappa \subseteq K \setminus (K \dot{-}^h K')$, then there exists a hybrid knowledge base K_1 such that $K \dot{-}^h K' \subseteq K_1 \subset K$ and $K_1 \not\models K'$ but $K_1 \cup \kappa \models K'$

($\dot{-}^h$4b) If it holds for all $\kappa \subseteq K$ that $\kappa \not\models K'$ iff $\kappa \not\models K''$, then $K \dot{-}^h K' = K \dot{-}^h K''$

($\dot{-}^h$5b) If $K \not\models K'$, then $K \dot{-}^h K' = K$

($\dot{-}^h$6b) If $K' \equiv K''$, then $K \dot{-}^h K' = K \dot{-}^h K''$

($\dot{-}^h$7b) $K \dot{-}^h (K' +^h K'') = K \dot{-}^h K'$ or $K \dot{-}^h (K' +^h K'') = K \dot{-}^h K''$ or $K \dot{-}^h (K' +^h K'') = (K \dot{-}^h K') \cap (K \dot{-}^h K'')$

($\dot{-}^h$8b) If $\kappa \subseteq K \setminus (K \dot{-}^h K')$, then $QHT(K \dot{-}^h K') \not\subseteq QHT(K') \cup QHT(\kappa)$

5

Partial Meet Belief Change

Inspired by our motivation, we begin in this section with defining belief change operators for logic programs that preserve syntactic information, in order to take into account information on the program-level as well as on the rule-level. We adapt the idea of a partial meet construction from classic belief change and formulate revision and contraction operators for logic programs. We utilise the adaptations of the AGM and belief base revision and contraction postulates from the previous section to test the rationality of our operators.

5.1 Partial Meet Revision

As the basis for our construction of partial meet revision, we define a *compatible set* of some program with respect to another program as the dual of a remainder set (Alchourrón et al. 1985).

Definition 5.1 (Compatible Set)**.** Let $P, Q \in \mathcal{LP}_{\mathcal{A}}$. The set of *compatible sets of P with respect to Q* is

$$\mathbb{P}_Q = \{\, R \subseteq P \mid SE(R) \cap SE(Q) \neq \emptyset \text{ and for all } R' \text{ with } R \subset R' \subseteq P : SE(R') \cap SE(Q) = \emptyset \,\}.$$

Each compatible set is a maximal subset of P that is consistent with Q under SE semantics. Each is thus a candidate to be returned together with Q as the outcome of a revision. To determine exactly which candidate(s) to choose, we employ a *selection function*.

Definition 5.2 (Selection Function)**.** A *selection function* γ for a set S is a function such that:

1. $\mathbb{S} \subseteq 2^S$,
2. $\gamma(\mathbb{S}) \subseteq \mathbb{S}$, and
3. if $\mathbb{S} \neq \emptyset$, then $\gamma(\mathbb{S}) \neq \emptyset$.

We can now define *partial meet revision* for logic programs as the intersection of the selected compatible sets added to Q.

Definition 5.3 (Partial Meet Revision)**.** Let $P \in \mathcal{LP}_\mathcal{A}$ and γ be a selection function for P. A *partial meet revision operator* $*_\gamma$ for P is defined such that for any $Q \in \mathcal{LP}_\mathcal{A}$:

$$P *_\gamma Q = \begin{cases} P + Q & \text{if } Q \text{ is not satisfiable,} \\ \bigcap \gamma(\mathbb{P}_Q) + Q & \text{otherwise.} \end{cases}$$

We illustrate the revision operation by an example. In the examples throughout this paper, we assume that the underlying language contains only the symbols that occur in the programs or that are otherwise mentioned explicitly.

Example 5.4. Let $P = \{\, a.,\ b \leftarrow a. \,\}$ and $Q = \{\, \bot \leftarrow a. \,\}$. We have $\mathbb{P}_Q = \{\, \{\, b \leftarrow a. \,\} \,\} = \gamma(\mathbb{P}_Q)$, for any selection function γ, and thus $P *_\gamma Q = \{\, b \leftarrow a.,\ \bot \leftarrow a. \,\}$. ■

The following theorem states which AGM revision postulates the revision operator $*_\gamma$ satisfies.

Theorem 5.5. The revision operator $*_\gamma$ satisfies $(*1)$–$(*6)$.

As in the classic case, to satisfy the supplementary postulates we would need to place further restrictions on the selection function.

Definition 5.6 (Relational Selection Function)**.** A *transitively relational selection function* γ' for a set S is determined by a transitive relation $\trianglelefteq$ over 2^S such that $\mathbb{S} \subseteq 2^S$ and:

$$\gamma'(\mathbb{S}) = \begin{cases} \emptyset & \text{iff } \mathbb{S} = \emptyset, \\ \{\, R \in \mathbb{S} \mid R' \trianglelefteq R \text{ for all } R' \in \mathbb{S} \,\} & \text{otherwise.} \end{cases}$$

The relation $\trianglelefteq$ is *maximised* iff for all $R, R' \in 2^S$: $R \subset R'$ implies $R \triangleleft R'$.

Yet, even if we make these additional restrictions on a selection function γ', postulates $(*7)$ and $(*8)$ are not satisfied by $*_{\gamma'}$, as shown respectively in the next two examples. For legibility, we confine the examples to abstract notation.

Example 5.7. Let $P = \{r_1, r_2, r_3\}$ with $SE(r_1) = \{B, C\}$, $SE(r_2) = \{A, C\}$, $SE(r_3) = \{A, B, C\}$, and $\{r_1, r_3\} \trianglelefteq \{r_2, r_3\} \trianglelefteq \{r_1, r_3\}$. If $SE(Q) = \{A, B\}$ and $SE(Q + R) = \{A\}$, then $\mathbb{P}_Q = \{\{r_1, r_3\}, \{r_2, r_3\}\}$ and thus $\bigcap \gamma'(\mathbb{P}_Q) = \{r_3\}$. On the other hand, we have $\mathbb{P}_{Q+R} = \{\{r_2, r_3\}\} = \bigcap \gamma'(\mathbb{P}_{Q+R})$. This means $(P *_{\gamma'} Q) + R = \{r_3\} \cup Q \cup R$ while $P *_{\gamma'} (Q + R) = \{r_2, r_3\} \cup Q \cup R$. ■

Example 5.8. Let $P = \{r_1, r_2, r_3, r_4, r_5\}$ with $SE(r_1) = \{A, B, C, D, E\}$, $SE(r_2) = \{A, B, C, E\}$, $SE(r_3) = \{A, E\}$, $SE(r_4) = \{B, E\}$, $SE(r_5) = \{D, E\}$, and $\{r_1, r_2\} \triangleleft \{r_1, r_5\} \triangleleft \{r_1, r_2, r_3\}, \{r_1, r_2, r_4\}$. If $SE(Q) = \{A, B, C, D\}$ and $SE(Q + R) = \{C, D\}$, then $\mathbb{P}_Q = \{\{r_1, r_2, r_3\}, \{r_1, r_2, r_4\}, \{r_1, r_5\}\}$ and $\mathbb{P}_{Q+R} = \{\{r_1, r_2\}, \{r_1, r_5\}\}$. It follows from $\gamma'(\mathbb{P}_Q) = \{\{r_1, r_2, r_3\}, \{r_1, r_2, r_4\}\}$ that $\bigcap \gamma'(\mathbb{P}_Q) = \{r_1, r_2\}$, while $\gamma'(\mathbb{P}_{Q+R}) = \{\{r_1, r_5\}\} = \bigcap \gamma'(\mathbb{P}_{Q+R})$. Therefore, $(P *_{\gamma'} Q) + R = \{r_1, r_2\} \cup Q \cup R \not\subseteq \{r_1, r_5\} \cup Q \cup R = P *_{\gamma'} (Q + R)$. Note that $SE((P *_{\gamma'} Q) + R) = SE(\bigcap \gamma'(\mathbb{P}_Q)) \cap SE(Q + R) = \{A, B, C, E\} \cap \{C, D\} = \{C\} \neq \emptyset$. ■

Our partial meet revision operator does not satisfy the entire set of AGM revision postulates, only the subset of basic postulates $(*1)$–$(*6)$. This stands in contrast to the result in classical logics, according to which any partial meet revision operator is characterised by the set of basic and supplementary revision postulates. However, that result holds for logically closed belief sets, and when we consider our partial meet revision operator $*_\gamma$ in the light of the belief base framework, we obtain the following representation theorem.

Lemma 5.9. Let $P, Q \in \mathcal{LP}_\mathcal{A}$ and γ be a selection function for P. If $r \in P \cap Q$, then $r \in \bigcap \gamma(\mathbb{P}_Q)$.

Theorem 5.10. Let $P \in \mathcal{LP}_{\mathcal{A}}$. An operator $*_\gamma$ is a partial meet revision operator for P generated by some selection function γ for P iff $*_\gamma$ satisfies ($*$1b)–($*$5b).

5.2 Partial Meet Contraction

Having defined a revision operator, we now turn to the case of belief contraction. In line with classic belief change, the contraction of a program P by a program Q should eliminate from P all those beliefs from which Q can be derived. We use the complement of $SE(Q)$, $\overline{SE(Q)}$, to determine all maximal subsets of P that do not imply Q, denoted as

$$\mathbb{P}^-_Q = \{\, R \subseteq P \mid SE(R) \cap \overline{SE(Q)} \neq \emptyset \text{ and for all } R' \text{ with } R \subset R' \subseteq P : SE(R') \cap \overline{SE(Q)} = \emptyset \,\}.$$

Definition 5.11 (Partial Meet Contraction)**.** Let $P \in \mathcal{LP}_{\mathcal{A}}$ and γ be a selection function for P. A *partial meet contraction operator* $\dot{-}_\gamma$ for P is defined such that for any $Q \in \mathcal{LP}_{\mathcal{A}}$:

$$P \dot{-}_\gamma Q = \begin{cases} P & \text{if } \not\models_s Q, \\ \bigcap \gamma(\mathbb{P}^-_Q) & \text{otherwise.} \end{cases}$$

Example 5.12. Let $P = \{\, a., b \leftarrow a. \,\}$ and $Q = \{\, a \leftarrow b. \,\}$. Since $\overline{SE(Q)} = \{(\emptyset, b), (b, b), (b, ab)\}$, $SE(\{\, a. \,\}) = \{(a, a), (a, ab), (ab, ab)\}$, and $SE(\{\, b \leftarrow a. \,\}) = \{(\emptyset, \emptyset), (\emptyset, b), (b, b), (\emptyset, ab), (b, ab), (ab, ab)\}$, it follows that $\mathbb{P}^-_Q = \{\, \{\, b \leftarrow a. \,\} \,\} = \gamma(\mathbb{P}^-_Q)$, for any selection function γ, and thus we obtain $P \dot{-}_\gamma Q = \{\, b \leftarrow a. \,\}$. ■

The next theorem lists the AGM contraction postulates that are fulfilled by $\dot{-}_\gamma$.

Theorem 5.13. The contraction operator $\dot{-}_\gamma$ satisfies ($\dot{-}$1)–($\dot{-}$4) and ($\dot{-}$6).

It is easy to see from Example 5.12 above that $\dot{-}_\gamma$ does not satisfy ($\dot{-}$5).

In the AGM framework, it is sufficient that the selection function γ is determined by a transitive relation so that $\ominus$ satisfies ($\ominus$7). Here, we require the relation $\trianglelefteq$ to be maximised as well to guarantee satisfaction of ($\dot{-}$7). This is in line with the result for partial meet base contractions, which also requires

the underlying selection function to be determined by a maximised transitive relation in order to satisfy such a property (Hansson 1993).

Lemma 5.14. Let $P \in \mathcal{LP}_{\mathcal{A}}$. For any $Q, R \in \mathcal{LP}_{\mathcal{A}}$, it holds that $\mathbb{P}^{-}_{Q+R} \subseteq \mathbb{P}^{-}_{Q} \cup \mathbb{P}^{-}_{R}$.

Lemma 5.15. Let $P \in \mathcal{LP}_{\mathcal{A}}$ and γ' be determined by a maximised transitive relation. For any $Q, R \in \mathcal{LP}_{\mathcal{A}}$, it holds that $\gamma'(\mathbb{P}^{-}_{Q+R}) \subseteq \gamma'(\mathbb{P}^{-}_{Q}) \cup \gamma'(\mathbb{P}^{-}_{R})$.

Theorem 5.16. Let γ' be determined by a maximised transitive relation. The contraction operator $\dot{-}_{\gamma'}$ satisfies ($\dot{-}$7).

The following example demonstrates that $\dot{-}_{\gamma'}$ does not satisfy ($\dot{-}$8).

Example 5.17. Consider again P and $\trianglelefteq$ from Example 5.8. If $SE(Q) = \{A, B, E\}$ and $SE(Q + R) = \{E\}$, then $\overline{SE(Q)} = \{C, D\}$ and $\overline{SE(Q + R)} = \{A, B, C, D\}$. We obtain $\mathbb{P}^{-}_{Q+R} = \{\{r_1, r_2, r_3\}, \{r_1, r_2, r_4\}, \{r_1, r_5\}\}$ and $\mathbb{P}^{-}_{Q} = \{\{r_1, r_2\}, \{r_1, r_5\}\}$. It follows from $\gamma'(\mathbb{P}^{-}_{Q+R}) = \{\{r_1, r_2, r_3\}, \{r_1, r_2, r_4\}\}$ that $\bigcap \gamma'(\mathbb{P}^{-}_{Q+R}) = \{r_1, r_2\}$, while $\gamma'(\mathbb{P}^{-}_{Q}) = \{\{r_1, r_5\}\}$ and thus $\bigcap \gamma'(\mathbb{P}^{-}_{Q}) = \{r_1, r_5\}$. Therefore, $P \dot{-}_{\gamma'} (Q + R) = \{r_1, r_2\} \not\subseteq \{r_1, r_5\} = P \dot{-}_{\gamma'} Q$. Note that $SE(P \dot{-}_{\gamma'} (Q + R)) = SE(\bigcap \gamma'(\mathbb{P}^{-}_{Q+R})) = \{A, B, C, E\} \not\subseteq \{A, B, E\} = SE(Q)$. ■

As in the case of our revision operator earlier, our partial meet contraction operator does not properly align with the AGM framework, but the following representation theorem holds for the contraction operator $\dot{-}_{\gamma}$ with respect to the belief base postulates.

Theorem 5.18. Let $P \in \mathcal{LP}_{\mathcal{A}}$. An operator $\dot{-}_{\gamma}$ is a partial meet contraction operator for P generated by some selection function γ for P iff $\dot{-}_{\gamma}$ satisfies ($\dot{-}$1b)–($\dot{-}$4b).

Besides the representation theorem above via ($\dot{-}$1b)–($\dot{-}$4b), we have the following additional properties of $\dot{-}_{\gamma}$ regarding the remaining belief base contraction postulates ($\dot{-}$5b)–($\dot{-}$8b).

Proposition 5.19. The contraction operator $\dot{-}_{\gamma}$ satisfies ($\dot{-}$5b), ($\dot{-}$6b), and ($\dot{-}$8b).

The next example illustrates that $\dot{-}_{\gamma}$ does not satisfy ($\dot{-}$7b).

Example 5.20. Let $P = \{r_1, r_2, r_3, r_4\}$ with $SE(r_1) = \{A, B, C, D, E\}$, $SE(r_2) = \{A, B, E\}$, $SE(r_3) = \{A, C, E\}$, and $SE(r_4) = \{C, D, E\}$. If $SE(Q) = \{A, D, E\}$ and $SE(R) = \{B, C, E\}$, then $\overline{SE(Q)} = \{B, C\}$, $\overline{SE(R)} = \{A, D\}$, and $\overline{SE(Q+R)} = \{A, B, C, D\}$. We thus have $\mathbb{P}^-_Q = \{\{r_1, r_2\}, \{r_1, r_3, r_4\}\}$, $\mathbb{P}^-_R = \{\{r_1, r_2, r_3\}, \{r_1, r_4\}\}$, and $\mathbb{P}^-_{Q+R} = \{\{r_1, r_2, r_3\}, \{r_1, r_3, r_4\}\}$. Let $\gamma(\mathbb{P}^-_Q) = \mathbb{P}^-_Q$, $\gamma(\mathbb{P}^-_R) = \mathbb{P}^-_R$, and $\gamma(\mathbb{P}^-_{Q+R}) = \mathbb{P}^-_{Q+R}$. It then follows that $\bigcap\gamma(\mathbb{P}^-_Q) = \{r_1\}$, $\bigcap\gamma(\mathbb{P}^-_R) = \{r_1\}$, and $\bigcap\gamma(\mathbb{P}^-_{Q+R}) = \{r_1, r_3\}$. Therefore, $P \dot{-}_\gamma (Q + R) \neq P \dot{-}_\gamma Q = P \dot{-}_\gamma R = (P \dot{-}_\gamma Q) \cap (P \dot{-}_\gamma R)$.

We now formalise the connection between partial meet revision and partial meet contraction with the help of the Levi and Harper identities as given in Definition 4.2.

Proposition 5.21. Let $P \in \mathcal{LP}_\mathcal{A}$, γ be a selection function for P, and $*$ an operator for P such that for any $Q \in \mathcal{LP}_\mathcal{A}$: $P * Q = (P \dot{-}_\gamma \overline{Q}) + Q$. Then $P * Q = P *_\gamma Q$.

Proposition 5.22. Let $P \in \mathcal{LP}_\mathcal{A}$, γ be a selection function for P, and $\dot{-}$ an operator for P such that for any $Q \in \mathcal{LP}_\mathcal{A}$: $P \dot{-} Q = P \cap (P *_\gamma \overline{Q})$. Then $P \dot{-} Q = P \dot{-}_\gamma Q$.

5.3 Localised Belief Change

While the definitions of our partial meet revision and contraction operators are based on classic declarative constructions, such formulations may not be optimal for practical implementations. In particular, the formation of a set of compatible sets to conduct a partial meet revision or contraction requires that all possible combinations of all rules in a program are evaluated with respect to their sets of SE models. When dealing with logic programs that contain a large number of rules, where only a small number of them are actually affected by the change operation, this procedure entails unreasonable costs. In this section, we present an algorithm to minimise these costs. We begin by identifying the subsets of a program, called *modules*, relevant to another program.

Definition 5.23 (Module)**.** Let $P \in \mathcal{LP}_\mathcal{A}$ and $a \in \mathcal{A}$. For any rule $r \in P$ with $a \in At(r)$, we recursively construct $M(P)^r_i|_a$ as

$$r \cup \{\, r' \in P \mid At(r') \cap \big(At(r) \cup At(M(P)^r_{i-1}|_a)\big) \setminus a \neq \emptyset \,\}$$

for $i > 0$ and $M(P)^r_0|_a = \emptyset$.

Since P is finite and $M(P)^r_i|_a$ is monotonic with respect to i, the sequence $\bigcup_{i=0}^{\infty} M(P)^r_i|_a$ will reach a fixpoint. We denote the fixpoint by $M(P)^r|_a$ and call it the *module of P related to r including a* (or the *r-module including a*, if P is clear from the context).

Example 5.24. Let r_1: $a.$, r_2: $b \leftarrow a.$, r_3: $c \leftarrow not\, b.$, and $P = \{r_1, r_2, r_3\}$. The modules that can be constructed from P are: $M(P)^{r_1}|_a = \{r_1\}, M(P)^{r_2}|_a = \{r_2, r_3\}, M(P)^{r_2}|_b = \{r_1, r_2\}, M(P)^{r_3}|_b = \{r_3\}$, and $M(P)^{r_3}|_c = \{r_1, r_2, r_3\}$. ■

Starting with a given atom a and a given rule r from P, the recursive definition first finds all rules in P that share atoms with r except for a. Then it finds all rules in P that share atoms with r or any of the rules found in the first step except for a, and so on. It does not matter whether atoms appear in the head or the body of a rule, or whether they occur with or without default negation. The resulting module is the collection of rules in P that are related to r through shared atoms. The reason for excluding a will become clear after the following definition of a set of relevant modules.

Definition 5.25 (Relevant Module)**.** Let $P \in \mathcal{LP}_{\mathcal{A}}$. Given an atom $a \in \mathcal{A}$, we define the *set of all modules of P including a* as:

$$\mathcal{M}(P)|_a = \{\, M(P)^r|_a \mid r \in P \text{ and } a \in At(r) \,\}.$$

Given $Q \in \mathcal{LP}_{\mathcal{A}}$, we define the *set of all modules of P relevant to Q* as:

$$\mathcal{M}(P)|_Q = \{\, M(P)^r|_a \mid r \in P \text{ and } a \in At(r) \cap At(Q) \,\}.$$

Essentially, the definition of a set of modules extracts those rules from a program that may be affected during a revision or contraction by another program. It thus aims for the same goal as the language-splitting technique in propositional logic (Parikh 1999), which splits a knowledge base into several partitions either relevant or irrelevant to a belief change. However, a distinct feature in the previous definitions is the construction of a module based on *each* rule in which a certain atom occurs. This feature allows us to get a closer look at which rules may conflict with some given information. Consider the program $\{\, a \leftarrow b., \bot \leftarrow b. \,\}$. If we were to add the information that "b holds" to this program, it would conflict with the latter rule but not with

the first one. By creating a module for each occurrence of b, we split the program into two modules (one for each rule) and can assess the compatibility of each module with the new information separately. This also separates our approach from the method to compute *compartments* (Hansson & Wassermann 2002, Wassermann 2000). That method assumes a graph representation of a belief base, where each sentence of the belief base is a node and edges connect sentences that share at least one atom. The parts of a belief base relevant to a given sentence ϕ for a change operation are the sentences that can be reached from ϕ. Thus, it does not distinguish between occurrences of ϕ as in our method. Furthermore, our method constructs modules for each individual atom occurring in Q and thus ensures that we are dealing with minimal units of P in a change operation. Obviously, a module may not be unique to a certain rule or a certain given atom so that modules may overlap or coincide.

We say that a set of rules R *conflicts* with some program Q if $SE(R) \cap SE(Q) = \emptyset$. All rules of P that conflict with Q are included in some module or combination of modules from $\mathcal{M}(P)|_Q$.

Proposition 5.26. Let $P, Q \in \mathcal{LP}_\mathcal{A}$ and $SE(P) \neq \emptyset \neq SE(Q)$. For any $R \subseteq P$, if $SE(R) \cap SE(Q) = \emptyset$ and for all $R' \subset R : SE(R') \cap SE(Q) \neq \emptyset$, then there exists $\mathbb{M} \in 2^{\mathcal{M}(P)|_Q}$ such that $R \subseteq \bigcup \mathbb{M}$.

Corollary 5.27. Let $P, Q \in \mathcal{LP}_\mathcal{A}$ and $SE(P) \neq \emptyset$. Then $SE(P) \cap SE(Q) = \emptyset$ if and only if $SE\left(\bigcup \mathcal{M}(P)|_Q\right) \cap SE(Q) = \emptyset$.

We are now ready to introduce an optimisation algorithm for logic program revision and contraction based on modules. Algorithm 1 resolves potential conflicts for all possible combinations of modules by applying revision or contraction on a modular level. It performs a bottom-up construction by first taking all 1-combinations (singleton sets of modules) of $\mathcal{M}$ and substituting a module with its changed version if they are not the same. It then takes all 2-combinations of $\mathcal{M}$, which may now contain some changed modules, and replaces each module of the combination with the changed version of the combination if required. Replacing *each* module of a combination with the outcome guarantees that the algorithm considers all possible combinations. The algorithm terminates after handling the combination of all modules in $\mathcal{M}$. The algorithm performs $|\mathcal{M}|^{|\mathcal{M}|/2}$ operations in the worst case, so its complexity is exponential. Since the formation of $\mathbb{P}_Q$ or $\mathbb{P}_Q^-$ requires $|2^P|^{|2^P|/2}$ operations in the worst case, the algorithm performs better whenever $|\mathcal{M}|$ is less than $|2^P|$.

Algorithm 1: ModChange

Input: a set $\mathcal{M}$ of modules, an operator $\circ$, a program Q
Output: the set $\mathcal{M}$ of changed modules

```
1  n ← 1;
2  while n ≤ |M| do
3  |  foreach 𝕄 ⊆ M such that |𝕄| = n do
4  |  |  if ∘ is a revision operator and SE(⋃𝕄) ∩ SE(Q) = ∅ then
5  |  |  |  foreach M ∈ 𝕄 do
6  |  |  |  |  replace M with (⋃𝕄 ∘ Q) \ Q in M;
7  |  |  |  end
8  |  |  else if ∘ is a contraction operator and SE(⋃𝕄) ∩ SE(Q)‾ = ∅
   |  |  |  then
9  |  |  |  foreach M ∈ 𝕄 do
10 |  |  |  |  replace M with ⋃𝕄 ∘ Q in M;
11 |  |  |  end
12 |  end
13 |  n ← n + 1;
14 end
15 return M;
```

The next theorem states that the algorithm ModChange reduces a partial meet revision or contraction operation on a logic program to the revision or contraction operation on the relevant subsets of that program, given a suitable selection function γ.

Theorem 5.28. For any $P, Q \in \mathcal{LP}_{\mathcal{A}}$, let $P \setminus \mathcal{M}(P)|_Q = \{\, r \in P \mid \forall M \in \mathcal{M}(P)|_Q : r \notin M \,\}$ and $\mathcal{M}(P)|_Q^{\circ}$ denote the output of Algorithm 1 for the inputs $\mathcal{M}(P)|_Q$, $\circ \in \{*_\gamma, \dot{-}_\gamma\}$, and Q. Then $P *_\gamma Q = P \setminus \mathcal{M}(P)|_Q + \bigcup \mathcal{M}(P)|_Q^{*_\gamma} + Q$ (or $P \dot{-}_\gamma Q = P \setminus \mathcal{M}(P)|_Q + \bigcup \mathcal{M}(P)|_Q^{\dot{-}_\gamma}$, respectively) for some selection function γ for P.

5.4 Discussion

In this chapter, we presented revision and contraction operators to execute belief change actions on logic programs. By using the monotonic SE model semantics and the notion of a compatible set, we were able to directly adapt a partial meet construction from classical logic. The advantage of our approach is that it performs belief change on the semantic level with respect to the SE models of programs, while at the same time preserving the syntactic information represented by their rules. We showed via representation theorems

that our operators fit tightly into the belief base framework. Furthermore, we showed that partial meet revision can be characterised in terms of partial meet contraction and vice versa, by applying the Levi and Harper identities. Finally, we developed a module-based algorithm that prunes a partial meet revision or contraction operation to performing the change only on the relevant subsets of a program.

Our work was motivated by identifying some shortcomings of the distance-based approach to logic program revision (Delgrande, Schaub, Tompits & Woltran 2013). We will illustrate later in Chapter 8 in detail how our partial meet revision operator addresses these shortcomings and how it compares to the distance-based revision operator with respect to the AGM and belief base revision postulates. To the best of our knowledge, ours is the first definition of a contraction operator for logic programs and the first adaptation of a partial meet construction to a nonmonotonic formalism.

6

Ensconcement and Entrenchment Belief Change in Logic Programs

In this chapter, we present two further approaches to belief change in logic programs. They are based on an ordering over the beliefs expressed by a program. We adapt the notions of *ensconcements* (Williams 1994) and *entrenchments* (Gärdenfors & Makinson 1988) from classic belief change for this purpose, which capture the intuition that some beliefs are more important to an agent than other beliefs. Ensconcements are applicable to belief change in knowledge bases that are not logically closed, which makes them a good choice for our adaptation to logic programs. Entrenchments, on the other hand, were originally proposed for knowledge bases that are closed under logical consequence, and we will investigate their suitability to define belief change in logic programs here.

In line with our approach of the previous chapter, our point of departure is to view an agent's beliefs as the rules and combinations of rules that belong to a program. We begin by defining an ensconcement as an order over rules of a program and define a revision and a contraction operator based on such an order. We evaluate our operators with respect to the AGM and belief

base postulates and characterise ensconcement revision in terms of ensconcement contraction and vice versa via the Levi and Harper identities. We show that ensconcement revision and contraction fully capture partial meet revision and contraction, respectively, but the converse property does not hold. Furthermore, we demonstrate that the outcomes of ensconcement revision or contraction operations are not influenced if we construct an ensconcement over the subsets of a program instead of over the rules of that program. To conclude our study of ensconcements, we show that Algorithm 1 can also be used to compute ensconcement revision or contraction operations.

We then turn to adapting entrenchments, where we divert from the original definition which implicitly requires all elements of a language to be ordered. Instead, we create an order only over rules and their combinations explicitly present in a program as a means to minimise costs. Based on such entrenchments, we present a contraction operator and via the Levi identity a revision operator, and evaluate their properties in light of the AGM framework. To the best of our knowledge, this is the first definition of ensconcements and entrenchments for a nonmonotonic formalism.

6.1 Ensconcement Revision

To represent an ordering over the beliefs contained in a logic program, we begin by defining an *ensconcement* relation for logic programs as follows.

Definition 6.1 (Ensconcement)**.** Let $P \in \mathcal{LP}_{\mathcal{A}}$. An *ensconcement associated with* P is any total preorder $\preceq$ on P that satisfies the following conditions:

($\preceq$1) For any $r \in P$: $SE(\{\, r' \in P \setminus \{r\} \mid r \preceq r' \,\}) \not\subset SE(r)$

($\preceq$2) For any $r, r' \in P$: $r \preceq r' \preceq r$ iff $\{r\} \equiv_s \{r'\}$

Per this definition, an ensconcement associated with a logic program P is simply an ordering over the rules occurring in P. With P representing our entire set of beliefs, an ensconcement enables us to sort rules of P, which form our individual beliefs, hierarchically by their epistemic importance, or in other words, by how willing we are to give up one belief over another. Informally, $r \prec r'$ means that the beliefs represented by r' are more important to us than the beliefs represented by r.

Condition ($\preceq$1) states that the set of SE models of any rule or combination of rules at least as ensconced as a given rule r may not be a proper subset of the set of SE models of r. Condition ($\preceq$2) requires that strongly equivalent rules are equally ensconced. Condition ($\preceq$1) is formulated slightly stronger than Condition ($\preccurlyeq$1) (see Section 2.4) from the original definition (Williams 1994). Condition ($\preccurlyeq$1) allows a sentence ψ, that implies a sentence ϕ without being equivalent to ϕ, to be placed on the same ensconcement level as ϕ. In contrast, Condition ($\preceq$1) prohibits strict implication on the same ensconcement level. For instance, given rules $a.$, $a \leftarrow b.$, and $a; b.$ contained in some program, both $a \leftarrow b.$ and $a; b.$ must be strictly more ensconced than $a.$ according to Condition ($\preceq$1), whereas in a direct adaptation of Condition ($\preccurlyeq$1) at least one of the two rules $a \leftarrow b.$ and $a; b.$ would have to be equally ensconced as $a.$. The merit of this additional restriction will become evident shortly, when we show some examples of applying an ensconcement to perform revision operations in Chapter 8. The idea behind Condition ($\preccurlyeq$2) of the original definition is that any tautologies must be most ensconced, a requirement that is automatically captured in our Condition ($\preceq$1).

During a revision operation, new information from a program Q is added to an initial belief state in the form of a program P, and some beliefs from P have to be given up to achieve a consistent outcome. When the beliefs in P are ordered by an ensconcement, we can introduce the notion of a *cut* to determine the specific level in the ensconcement where all beliefs on and above this level are consistent with the revising program. Since an ensconcement $\preceq$ associated with P is a relation over all rules of P, when we write $r \preceq r'$, we implicitly mean $r \in P$ and $r' \in P$.

Definition 6.2 (Cut)**.** Let $P, Q \in \mathcal{LP}_{\mathcal{A}}$ and $\preceq$ be an ensconcement associated with P. The *(proper) cut of P for Q*, written $cut_{\preceq}(Q)$, is defined as

$$cut_{\preceq}(Q) = \{\, r \in P \mid SE(\{\, r' \in P \mid r \preceq r' \,\}) \cap SE(Q) \neq \emptyset \,\}.$$

Some interesting properties of the cut are listed below.

Lemma 6.3. Let $P, Q, R \in \mathcal{LP}_{\mathcal{A}}$ and $\preceq$ be an ensconcement associated with P.

a) If $P + Q$ is satisfiable, then $cut_{\preceq}(Q) = P$.

b) If Q is satisfiable, then $cut_{\preceq}(Q) + Q$ is satisfiable.

c) If Q is not satisfiable, then $cut_{\preceq}(Q) = \emptyset$.

d) If $Q \models_s R$, then $cut_{\preceq}(Q) \subseteq cut_{\preceq}(R)$.

e) $cut_{\preceq}(Q + R) \subseteq cut_{\preceq}(Q)$.

f) If $cut_{\preceq}(Q) \models_s R$, then $cut_{\preceq}(Q + R) = cut_{\preceq}(Q)$.

A cut is the principal element for the following definition of an ensconcement revision operator.

Definition 6.4 (Ensconcement Revision)**.** Let $P \in \mathcal{LP}_{\mathcal{A}}$ and $\preceq$ be an ensconcement associated with P. An *ensconcement revision operator* $*_{\preceq}$ for P is defined such that for any $Q \in \mathcal{LP}_{\mathcal{A}}$:

$$P *_{\preceq} Q = \begin{cases} P + Q & \text{if } Q \text{ is not satisfiable,} \\ \{\, r \in P \mid SE(cut_{\preceq}(Q)) \cap SE(Q) \subseteq SE(r) \,\} + Q & \text{otherwise.} \end{cases}$$

The revision operator $*_{\preceq}$ retains all elements of the cut. This is an obvious requirement since the cut contains our most firmly held beliefs which are entirely consistent with Q. In addition, any rule of P not in the cut that shares the same SE models with Q as the cut is retained as well. The example below illustrates the operation.

Example 6.5. Let $P = \{\, a.,\ a \leftarrow b.,\ b \leftarrow a. \,\}$ and $Q = \{\, \bot \leftarrow b. \,\}$. Figure 6.1 shows all possible ensconcements associated with P, with rules displayed at the top being more ensconced than rules at the bottom. We have the following results:

1. $cut_{\preceq_1}(Q) = \{\, a \leftarrow b.,\ b \leftarrow a. \,\}$ and $P *_{\preceq_1} Q = \{\, a \leftarrow b.,\ b \leftarrow a.,\ \bot \leftarrow b. \,\}$
2. $cut_{\preceq_2}(Q) = \{\, a \leftarrow b.,\ b \leftarrow a. \,\}$ and $P *_{\preceq_2} Q = \{\, a \leftarrow b.,\ b \leftarrow a.,\ \bot \leftarrow b. \,\}$
3. $cut_{\preceq_3}(Q) = \{\, a \leftarrow b.,\ b \leftarrow a. \,\}$ and $P *_{\preceq_3} Q = \{\, a \leftarrow b.,\ b \leftarrow a.,\ \bot \leftarrow b. \,\}$
4. $cut_{\preceq_4}(Q) = \{\, a \leftarrow b. \,\}$ and $P *_{\preceq_4} Q = \{\, a \leftarrow b.,\ \bot \leftarrow b. \,\}$
5. $cut_{\preceq_5}(Q) = \{\, a \leftarrow b.,\ a. \,\}$ and $P *_{\preceq_5} Q = \{\, a \leftarrow b.,\ a.,\ \bot \leftarrow b. \,\}$ ■

In Example 6.5, the belief expressed by the combination of rules $\{\, a. \,\}$ and $\{\, b \leftarrow a. \,\}$ is inconsistent with the new information $\{\, \bot \leftarrow b. \,\}$. Thus, at least one of these two rules must be discarded to reach a consistent belief state,

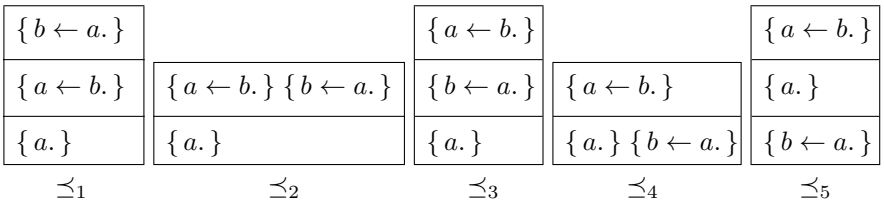

Figure 6.1: $\preceq_1$, $\preceq_2$, $\preceq_3$, $\preceq_4$, $\preceq_5$ of Example 6.5

while the rule $\{ a \leftarrow b. \}$ can be safely retained. The example shows that the revision operator $*_{\preceq}$ indeed retains $\{ a \leftarrow b. \}$ in all cases and discards one or both other rules depending on their ensconcement level. Whenever $\{ b \leftarrow a. \}$ is more ensconced than $\{ a. \}$, the latter is discarded and vice versa. Only when both rules are equally ensconced, that is, when we cannot make up our mind which the two beliefs we hold more firmly, the revision operator discards both.

We can see from the definition of $*_{\preceq}$ that the set of SE models of $P *_{\preceq} Q$ is exactly the set of SE models that are shared by $cut_{\preceq}(Q)$ and Q.

Proposition 6.6. Let $P, Q \in \mathcal{LP}_{\mathcal{A}}$ and $\preceq$ be an ensconcement associated with P. Then $SE(P *_{\preceq} Q) = SE(cut_{\preceq}(Q) + Q)$.

The next theorem states which of the adapted AGM revision postulates the revision operator $*_{\preceq}$ satisfies.

Theorem 6.7. The revision operator $*_{\preceq}$ satisfies $(*1)$–$(*6)$ and $(*8)$.

The revision operator $*_{\preceq}$ does not satisfy $(*7)$, as shown in the next example.

Example 6.8. Let $P = \{r_1, r_2, r_3\}$ with $SE(r_1) = \{B, C\}$, $SE(r_2) = \{A, C\}$, $SE(r_3) = \{A, B, C\}$, and $\preceq$ be an ensconcement associated with P such that $r_1 \preceq r_2 \preceq r_1 \prec r_3$. If $SE(Q) = \{A, B\}$ and $SE(Q + R) = \{A\}$, then $cut_{\preceq}(Q) = cut_{\preceq}(Q + R) = \{r_3\}$, yet $(P *_{\preceq} Q) + R = \{r_3\} \cup Q \cup R$ while $P *_{\preceq} (Q + R) = \{r_2, r_3\} \cup Q \cup R$. ■

Even though ensconcement revision was originally defined for belief bases, our ensconcement revision operator $*_{\preceq}$ satisfies the majority of AGM revision postulates for belief sets. Our operator does not satisfy any of the postulates that are unique to the belief base framework, as stated in the next theorem.

Theorem 6.9. The revision operator $*_{\preceq}$ satisfies $(*1b)$, $(*2b)$, and $(*5b)$.

In the following two examples, we illustrate that $*_{\preceq}$ does indeed not satisfy (*3b) and (*4b), respectively.

Example 6.10. Let $\mathcal{SE} = \{A, B, C, D\}$, $SE(Q) = \{A, B\}$, and $P = \{r_1, r_2, r_3\}$ with $SE(r_1) = \{C, D\}$, $SE(r_2) = \{B, C\}$, and $SE(r_3) = \{A, B, C\}$. If $\preceq$ is an ensconcement associated with P such that $r_1 \preceq r_2 \preceq r_1 \prec r_3$, then $cut_{\preceq}(Q) = \{r_3\}$ and $P *_{\preceq} Q = \{r_3\} + Q$. While $r_2 \in P \setminus (P *_{\preceq} Q)$, there exists no program P' such that $P *_{\preceq} Q \subseteq P' \subseteq P + Q$ and P' is satisfiable but $P' \cup \{r_2\}$ is not satisfiable. ■

Example 6.11. Let $\mathcal{SE} = \{A, B, C, D\}$, $SE(Q) = \{A, B\}$, $SE(R) = \{A\}$, and $P = \{r_1, r_2, r_3\}$ with $SE(r_1) = \{C, D\}$, $SE(r_2) = \{A, C\}$, and $SE(r_3) = \{A, B, C\}$. If $\preceq$ is an ensconcement associated with P such that $r_1 \preceq r_2 \preceq r_1 \prec r_3$, then it holds for any $P' \subseteq P$ that $P' + Q$ is satisfiable iff $P' + R$ is satisfiable. However, we have $cut_{\preceq}(Q) = \{r_3\}$ and $P \cap (P *_{\preceq} Q) = \{r_3\}$, while $cut_{\preceq}(R) = \{r_3\}$ and $P \cap (P *_{\preceq} R) = \{r_2, r_3\}$. ■

6.2 Ensconcement Contraction

We now use the concept of an ensconcement to present another contraction operator for logic programs. Analogous to revision, we first define for some $P, Q \in \mathcal{LP}_{\mathcal{A}}$ and an ensconcement $\preceq$ associated with P that

$$cut^{-}_{\preceq}(Q) = \{\, r \in P \mid SE(\{\, r' \in P \mid r \preceq r' \,\}) \cap \overline{SE(Q)} \neq \emptyset \,\}.$$

Some useful properties of $cut^{-}_{\preceq}(Q)$ are listed here.

Lemma 6.12. Let $P, Q, R \in \mathcal{LP}_{\mathcal{A}}$ and $\preceq$ be an ensconcement associated with P.

a) If $P \not\models_s Q$, then $cut^{-}_{\preceq}(Q) = P$.

b) If $\not\models_s Q$, then $cut^{-}_{\preceq}(Q) \not\models_s Q$.

c) If $\models_s Q$, then $cut^{-}_{\preceq}(Q) = \emptyset$.

d) If $Q \models_s R$, then $cut^{-}_{\preceq}(R) \subseteq cut^{-}_{\preceq}(Q)$.

e) $cut^{-}_{\preceq}(Q) \subseteq cut^{-}_{\preceq}(Q + R)$.

f) If $cut^{-}_{\preceq}(Q) \models_s R$, then $cut^{-}_{\preceq}(Q + R) = cut^{-}_{\preceq}(Q)$.

g) If $cut^{-}_{\preceq}(Q) \not\models_s R$, then $cut^{-}_{\preceq}(Q + R) = cut^{-}_{\preceq}(R)$.

Definition 6.13 (Ensconcement Contraction)**.** Let $P \in \mathcal{LP}_{\mathcal{A}}$ and $\preceq$ be an ensconcement associated with P. An *ensconcement contraction operator* $\dot{-}_{\preceq}$ for P is defined such that for any $Q \in \mathcal{LP}_{\mathcal{A}}$:

$$P \dot{-}_{\preceq} Q = \begin{cases} P & \text{if } \models_s Q, \\ \{\, r \in P \mid SE(cut^{-}_{\preceq}(Q)) \cap \overline{SE(Q)} \subseteq SE(r) \,\} & \text{otherwise.} \end{cases}$$

The contraction operator $\dot{-}_{\preceq}$ works in a dual way to the revision operator $*_{\preceq}$. It relies on $cut^{-}_{\preceq}(Q)$ to determine from which level upward in the ensconcement associated with P elements are retained in the operation, and adds any further parts of P that do not compromise the set of SE models of $cut^{-}_{\preceq}(Q)$ inconsistent with Q.

We can formalise the relationship between the SE models of $P \dot{-}_{\preceq} Q$ and $cut^{-}_{\preceq}(Q)$ as follows.

Proposition 6.14. Let $P, Q \in \mathcal{LP}_{\mathcal{A}}$ and $\preceq$ be an ensconcement associated with P. Then $SE(P \dot{-}_{\preceq} Q) \cap \overline{SE(Q)} = SE(cut^{-}_{\preceq}(Q)) \cap \overline{SE(Q)}$.

The contraction operator $\dot{-}_{\preceq}$ satisfies all AGM contraction postulates except Recovery.

Theorem 6.15. The contraction operator $\dot{-}_{\preceq}$ satisfies ($\dot{-}$1)–($\dot{-}$4) and ($\dot{-}$6)–($\dot{-}$8).

The next example shows why $\dot{-}_{\preceq}$ does not satisfy the Recovery postulate ($\dot{-}$5).

Example 6.16. Consider again P and Q from Example 5.12. For any ensconcement $\preceq$ associated with P, it holds that $P \dot{-}_{\preceq} Q = \{\, b \leftarrow a. \,\}$. Thus, $P = \{\, a.,\ b \leftarrow a. \,\} \not\subseteq \{\, b \leftarrow a.,\ a \leftarrow b. \,\} = (P \dot{-}_{\preceq} Q) + Q$. ■

The main reason for non-satisfaction of Recovery is that our ensconcement contraction operator $\dot{-}_{\preceq}$ operates on programs that are not logically closed, and Recovery is a key AGM postulate that characterises contractions of logically closed belief sets. On the other hand, our ensconcement contraction operator satisfies the same set of belief base postulates as its classic counterpart.

Theorem 6.17. The contraction operator $\dot{-}_{\preceq}$ satisfies (−1b), (−2b) and (−5b)–(−8b).

The following two examples demonstrate that the contraction operator $\dot{-}_{\preceq}$ does not satisfy ($\dot{-}$3b) and ($\dot{-}$4b), respectively.

Example 6.18. Let $\mathcal{SE} = \{A, B, C, D\}$, $SE(Q) = \{C\}$, and consider again P and $\preceq$ from Example 6.10. Then $cut^{-}_{\preceq}(Q) = \{r_3\} = P \dot{-}_{\preceq} Q$. While $r_2 \in P \setminus (P \dot{-}_{\preceq} Q)$, there exists no program P' such that $P \dot{-}_{\preceq} Q \subseteq P' \subseteq P$ and $P' \not\models_s Q$ but $P' \cup \{r_2\} \models_s Q$. ■

Example 6.19. Let $\mathcal{SE} = \{A, B, C, D\}$, $SE(Q) = \{C\}$, $SE(R) = \{B, C\}$, and consider again P and $\preceq$ from Example 6.11. Then it holds for any $P' \subseteq P$ that $P' \not\models_s Q$ iff $P' \not\models_s R$. However, we have $cut^{-}_{\preceq}(Q) = \{r_3\}$ and $P \dot{-}_{\preceq} Q = \{r_3\}$, while $cut^{-}_{\preceq}(R) = \{r_3\}$ and $P \dot{-}_{\preceq} R = \{r_2, r_3\}$. ■

As for our partial meet operators, the characterisation via Levi and Harper identities also holds for ensconcement revision and ensconcement contraction.

Proposition 6.20. Let $P \in \mathcal{LP}_{\mathcal{A}}$, $\preceq$ be an ensconcement associated with P, and $*$ an operator for P such that for any $Q \in \mathcal{LP}_{\mathcal{A}}$: $P * Q = (P \dot{-}_{\preceq} \overline{Q}) + Q$. Then $P * Q = P *_{\preceq} Q$.

Proposition 6.21. Let $P \in \mathcal{LP}_{\mathcal{A}}$, $\preceq$ be an ensconcement associated with P, and $\dot{-}$ an operator for P such that for any $Q \in \mathcal{LP}_{\mathcal{A}}$: $P \dot{-} Q = P \cap (P *_{\preceq} \overline{Q})$. Then $P \dot{-} Q = P \dot{-}_{\preceq} Q$.

6.3 Connections between Partial Meet and Ensconcement Belief Change

Having defined partial meet revision and contraction operators and ensconcement revision and contraction operators in the previous sections, we now establish the formal connections between them. We first relate partial meet revision to ensconcement revision and partial meet contraction to ensconcement contraction. We then investigate whether the granularity of ensconcements, that is, whether an ensconcement is defined over rules or subsets of a program, influences that relationship. Finally, we show that our algorithm that performs a partial meet revision or contraction operation only on the relevant subsets of a program can also be applied to ensconcement revision or contraction.

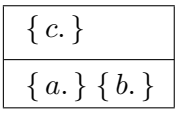

Figure 6.2: $\preceq$ of Example 6.23

6.3.1 Relating Partial Meet Operators to Ensconcement Operators

We already saw from the set of postulates that $*_{\preceq}$ and $*_{\gamma}$ satisfy, that partial meet revision and ensconcement revision share similar properties. In the following characterisation theorem we state the exact relationship between the two.

Theorem 6.22. Let $P, Q \in \mathcal{LP}_{\mathcal{A}}$. For any selection function γ, there exists an ensconcement $\preceq$ associated with P such that $P *_{\gamma} Q = P *_{\preceq} Q$.

Theorem 6.22 asserts that $*_{\gamma}$ can be characterised in terms of $*_{\preceq}$. The other direction is not possible, as shown in the example below.

Example 6.23. Let $P = \{ a., b., c. \}$, $Q = \{ \bot \leftarrow a. \}$, and $\preceq$ be the ensconcement associated with P as shown in Figure 6.2. It follows that $cut_{\preceq}(Q) = \{ c. \}$ and $P *_{\preceq} Q = \{ c., \bot \leftarrow a. \}$. Yet $\mathbb{P}_Q = \{ \{ b., c. \} \} = \gamma(\mathbb{P}_Q)$, for any selection function γ, so that $P *_{\gamma} Q = \{ b., c., \bot \leftarrow a. \}$. We have $P *_{\preceq} Q \neq P *_{\gamma} Q$. ■

On the one hand, the requirement $SE(cut_{\preceq}(Q)) \cap SE(Q) \subseteq SE(R)$ in Definition 6.4 requires any subset R of P that is not part of the cut to have *all* SE models shared by the cut and Q, in order to be included in the revision outcome. In the previous example, the SE models shared by the cut and Q are $(c, c), (c, bc)$, and (bc, bc). Since $(c, c) \notin SE(\{ b. \})$ (and also $(c, bc) \notin SE(\{ b. \})$), it follows that $\{ b. \} \not\subseteq P *_{\preceq} Q$. On the other hand, the definition of partial meet revision is based on compatible sets, which are required to be maximal and to share only a minimum of one SE model with Q (Definition 5.1). This requirement limits the result of a partial meet revision for this example to the one above, regardless of the the type of selection function employed.

We also find that the partial meet contraction operator $\dot{-}_{\gamma}$ can be characterised in terms of the ensconcement contraction operator $\dot{-}_{\preceq}$, formalised in the next theorem.

Theorem 6.24. Let $P, Q \in \mathcal{LP}_{\mathcal{A}}$. For any selection function γ, there exists an ensconcement $\preceq$ associated with P such that $P \dot{-}_{\gamma} Q = P \dot{-}_{\preceq} Q$.

The other direction of this theorem does not hold. Consider again P and $\preceq$ from Example 6.23 and let $Q = \{ a. \}$. It is easy to see that $P \dot{-}_{\gamma} Q \neq P \dot{-}_{\preceq} Q$, for any selection function γ.

6.3.2 Granularity of Ensconcements

For our partial meet construction, we determined the outcome of a revision or contraction operation by employing a function that selects among *subsets* of a program. For our ensconcement construction, we then used an ordering over individual rules of a program to select the *rules* to retain during a revision or contraction operation. Given the Characterisation Theorems 6.22 and 6.24 that hold only in one direction, it is worth investigating whether this difference in granularity, subsets or rules, plays a critical role in determining revision or contraction outcomes. To do so, we will now consider subsets as the objects of change for our ensconcement revision and contraction operators. We begin with the definition of an ensconcement over subsets of a program.

Definition 6.25 (Ensconcement over Subsets)**.** Given $P \in \mathcal{LP}_{\mathcal{A}}$, a *subset-ensconcement associated with* P is any total preorder $\preceq^R$ on 2^P that satisfies the following conditions:

($\preceq^R$1) For any $R \subseteq P$: $SE(\{ R' \subseteq P \setminus \{R\} \mid R \preceq R' \}) \not\subset SE(R)$

($\preceq^R$2) For any $R, R' \subseteq P$: $R \preceq R' \preceq R$ iff $\{R\} \equiv_s \{R'\}$

We define revision and contraction operators based on $\preceq^R$ as follows.

Definition 6.26 (Subset-Ensconcement Revision)**.** Let $P \in \mathcal{LP}_{\mathcal{A}}$ and $\preceq^R$ be a subset-ensconcement associated with P. A *subset-ensconcement revision operator* $*_{\preceq^R}$ for P is defined such that for any $Q \in \mathcal{LP}_{\mathcal{A}}$:

$$P*_{\preceq^R}Q = \begin{cases} P + Q & \text{if } Q \text{ is not satisfiable,} \\ \{ R \subseteq P \mid SE(cut_{\preceq^R}(Q)) \cap SE(Q) \subseteq SE(R) \} + Q & \text{otherwise,} \end{cases}$$

where $cut_{\preceq^R}(Q) = \{ R \subseteq P \mid SE(\{ R' \subseteq P \mid R \preceq R' \}) \cap SE(Q) \neq \emptyset \}$.

Definition 6.27 (Subset-Ensconcement Contraction)**.** Let $P \in \mathcal{LP}_{\mathcal{A}}$ and $\preceq^R$ be a subset-ensconcement associated with P. A *subset-ensconcement contrac-*

tion operator $\dot{-}_{\preceq^R}$ for P is defined such that for any $Q \in \mathcal{LP}_\mathcal{A}$:

$$P \dot{-}_{\preceq^R} Q = \begin{cases} P & \text{if } \models_s Q, \\ \{\, R \subseteq P \mid SE(cut^-_{\preceq^R}(Q)) \cap \overline{SE(Q)} \subseteq SE(R) \,\} & \text{otherwise,} \end{cases}$$

where $cut^-_{\preceq^R}(Q) = \{\, R \subseteq P \mid SE\,(\{\, R' \subseteq P \mid R \preceq R' \,\}) \cap \overline{SE(Q)} \neq \emptyset \,\}$.

The following theorem states that it does not matter whether an ensconcement over subsets of a program or only over the individual rules is used to determine a revision or contraction outcome, provided that the individual rules are ordered in the same way in both ensconcements.

Lemma 6.28. Let $\preceq^R$ be a subset-ensconcement associated with some $P \in \mathcal{LP}_\mathcal{A}$ and $R \subseteq P$. For any rule $r \in R$, it holds that $R \preceq^R \{r\}$.

Theorem 6.29. Let $P, Q \in \mathcal{LP}_\mathcal{A}$, $\preceq$ be an ensconcement associated with P, and $\preceq^R$ a subset-ensconcement associated with P such that $\{r\} \preceq^R \{r'\}$ iff $r \preceq r'$ for all $r, r' \in P$. Then $P *_\preceq Q = P *_{\preceq^R} Q$ (or $P \dot{-}_\preceq Q = P \dot{-}_{\preceq^R} Q$, alternatively).

6.3.3 Localised Belief Change

In Section 5.3, we defined the modules of a program that are relevant to a revision or contraction operation and introduced Algorithm 1, which reduces a partial meet revision or contraction to revising or contracting only the relevant modules of that program. We have just established in Theorems 6.22 and 6.24 that $*_\gamma$ and $\dot{-}_\gamma$ can be characterised in terms of $*_\preceq$ and $\dot{-}_\preceq$, respectively. Therefore, we can directly extend Theorem 5.28 to ensconcement revision and contraction.

Corollary 6.30. For any $P, Q \in \mathcal{LP}_\mathcal{A}$, let $P \setminus \mathcal{M}(P)|_Q = \{\, r \in P \mid \forall M \in \mathcal{M}(P)|_Q : r \notin M \,\}$ and $\mathcal{M}(P)|^\circ_Q$ denote the output of Algorithm 1 for the inputs $\mathcal{M}(P)|_Q$, $\circ \in \{*_\preceq, \dot{-}_\preceq\}$, and Q. Then $P *_\preceq Q = P \setminus \mathcal{M}(P)|_Q + \bigcup \mathcal{M}(P)|^{*_\preceq}_Q + Q$ (or $P \dot{-}_\preceq Q = P \setminus \mathcal{M}(P)|_Q + \bigcup \mathcal{M}(P)|^{\dot{-}_\preceq}_Q$, respectively) for some ensconcement $\preceq$ associated with P.

6.4 Entrenchment Contraction

In this section, we introduce an *entrenchment contraction operator* for logic programs. Under the classic definition (Gärdenfors & Makinson 1988), an *epistemic entrenchment* associated with some belief set is a relation on the entire language from which the belief set is constructed. Implicitly, this assumes that an agent is capable of specifying for every single belief expressible in the language how valuable it is over another belief. However, it is not realistic to be applied in practical scenarios with limited resources. In the following, we define an epistemic entrenchment associated with a logic program P as a relation only on the beliefs contained in P.

Definition 6.31 (Epistemic Entrenchment)**.** Given $P \in \mathcal{LP}_{\mathcal{A}}$, an *(epistemic) entrenchment associated with* P is any total order $\leq$ on 2^P that satisfies the following conditions:

($\leq$1) $\leq$ is transitive.

($\leq$2) If $X \models_s Y$, then $X \leq Y$.

($\leq$3) $X \leq X \cup Y$ or $Y \leq X \cup Y$.

Intuitively, by $X \leq Y$ we mean that Y is at least as entrenched as X, or in other words, an agent is more likely to give up its belief X than its belief Y. Apart from restricting an entrenchment relation to the beliefs explicitly contained in P, Conditions ($\leq$1)–($\leq$3) are adaptations from epistemic entrenchments in classical logics: ($\leq$1) ensures a transitive order, ($\leq$2) states that logically weaker programs are at least as entrenched as stronger ones and strongly equivalent programs are equally entrenched, and ($\leq$2)–($\leq$3) require the union of two programs to be equally entrenched as the least entrenched of the two. Conditions (EE4) and (EE5) from the original definition (see Section 2.4) are not adapted here. Condition (EE4) states that non-beliefs are least entrenched. This condition is not necessary here, since $\leq$ is a relation on 2^P, which means that all beliefs ordered via $\leq$ are implied by P. Condition (EE5) states that tautologies are most entrenched. Since an entrenchment is defined only over beliefs contained in P, most entrenched beliefs are not necessarily tautologies.

Note that Conditions ($\leq$1)–($\leq$3) merely specify necessary but not sufficient conditions to create an entrenchment ordering over the entire set of beliefs. In

cases where more than one epistemic entrenchment can be associated with a logic program, the agent's evaluation of its beliefs is required to determine the final structure of an entrenchment. It is important to point out that essentially only the rules of P need to be evaluated, because any combination of rules is automatically placed in the entrenchment due to ($\leq$2) and ($\leq$3).

While our definition facilitates an efficient construction of an epistemic entrenchment, it poses a problem for establishing a contraction operation that is built on such an entrenchment. Existing entrenchment contraction operators rely on comparing the entrenchment level of the beliefs in an initial belief state to the entrenchment level of the beliefs of be contracted. This requires that the beliefs to be contracted are covered by the entrenchment relation. In classical logics, a belief state is represented as a logically closed belief set K, which means that any beliefs implied by K that are to be contracted are found on some level in the entrenchment ordering associated with K. Logic programs, on the other hand, are not closed under logical consequence, so that beliefs to be contracted may not be part of the entrenchment relation defined above. We solve this problem by allowing an epistemic entrenchment associated with a logic program to be extended by adding to the entrenchment the beliefs to be contracted. Before we do so in the next definition, we introduce some notation. Given an entrenchment $\leq$ on $S \in \mathcal{LP}_{\mathcal{A}}$, by $\leq|_{S'}$ we denote the relation $\leq$ on $S' \subseteq S$, i.e., the relation $\leq$ restricted to S'.

Definition 6.32 (Extended Epistemic Entrenchment)**.** Let $P \in \mathcal{LP}_{\mathcal{A}}$ and $\leq$ be an epistemic entrenchment associated with P. For any $Q \in \mathcal{LP}_{\mathcal{A}}$, an *(epistemic) entrenchment associated with P plus Q* is any total order $\leq_Q$ on $2^P \cup 2^Q$ that satisfies the following conditions:

($\leq_Q$1) $\leq_Q$ is transitive.

($\leq_Q$2) If $X \models_s Y$, then $X \leq_Q Y$.

($\leq_Q$3) $X \leq_Q X \cup Y$ or $Y \leq_Q X \cup Y$.

($\leq_Q$4) If P is satisfiable and $X \in \mathcal{LP}_{\mathcal{A}}$, then $P \not\models_s X$ iff $X \leq_Q P$ and $P \not\leq_Q X$.

($\leq_Q$5) $\leq_Q|_{2^P} = \leq$ and for any $X \in 2^Q \backslash 2^P$, X is $\leq_Q$-minimal under satisfaction of ($\leq_Q$1)–($\leq_Q$4).

Condition ($\leq_Q$4) is necessary here because the relation $\leq_Q$ is on $2^P \cup 2^Q$, which means that it may include beliefs that are not implied by P. Condition ($\leq_Q$5) ensures that the entrenchment over P remains unchanged and that

any added beliefs are placed on the lowest level possible. With this definition of an extended epistemic entrenchment established we can now move forward with the construction of a contraction operator. A fundamental problem of adapting entrenchment contraction from classical logics to logic programs however is that the condition

$$\psi \in K \ominus_{\leqslant} \phi \text{ iff } \psi \in K \text{ and either } \vdash \phi \text{ or } \phi < \phi \vee \psi$$

relies upon the entrenchment level of a disjunction of formulas. There is no consensus on how to define disjunction for logic programs. Therefore, we follow the first-order conjunctive (FC) logic contraction approach (Zhuang, Wang, Wang & Delgrande 2015), which circumvents disjunctions by using critical formulas. We first adapt the concept of critical formulas to logic programs by defining *critical programs*. Given two programs P and Q, a critical program of some subset R of P is any subset R' of P such that each SE model of R' is an SE model of Q or R.

Definition 6.33 (Critical Program)**.** Let $P, Q \in \mathcal{LP}_{\mathcal{A}}$. The set of *critical programs of $R \subseteq P$ with respect to Q*, denoted $C_Q(R)$, is given by:

$$R' \in C_Q(R) \text{ iff } R' \subseteq P \text{ and } SE(R') \subseteq SE(Q) \cup SE(R).$$

Critical programs together with an extended epistemic entrenchment provide the foundation for our definition of an entrenchment contraction operator for logic programs as follows.

Definition 6.34 (Entrenchment Contraction)**.** Let $P, Q \in \mathcal{LP}_{\mathcal{A}}$ and $\leq_Q$ be an epistemic entrenchment associated with P plus Q. We define an operator $\dot{-}_{\leq_Q}$ as an *entrenchment contraction operator* for P such that

$$P \dot{-}_{\leq_Q} Q = \begin{cases} P & \text{if } \not\models_s Q, \\ \bigcup \{\, R \subseteq P \mid \exists R' \in C_Q(R) \text{ such that } Q <_Q R' \,\} & \text{otherwise.} \end{cases}$$

By Definition 6.34, a subset R of P is retained in the contraction whenever there exists a critical program of R that is strictly more entrenched than Q. We illustrate the operation in the following example.

Example 6.35. Let $P = \{\, a.,\ b \leftarrow a.,\ b. \,\}$ and $Q = \{\, a.,\ b. \,\}$. It follows that

$$C_Q(\{\, a. \,\}) = \{\, \{\, a. \,\}, \{\, a.,\ b \leftarrow a. \,\}, \{\, a.,\ b. \,\}, P \,\};$$

$$
\begin{aligned}
C_Q(\{\, b \leftarrow a. \,\}) &= \{\, \{\, b \leftarrow a. \,\}, \{\, b. \,\}, \{\, a.,\, b \leftarrow a. \,\}, \\
&\qquad \{\, a.,\, b. \,\}, \{\, b \leftarrow a., b. \,\}, P \,\}; \\
C_Q(\{\, b. \,\}) &= \{\, \{\, b. \,\}, \{\, a.,\, b \leftarrow a. \,\}, \{\, a.,\, b. \,\}, \{\, b \leftarrow a., b. \,\}, P \,\}; \\
C_Q(\{\, a.,\, b \leftarrow a. \,\}) &= \{\, \{\, a.,\, b \leftarrow a. \,\}, \{\, a.,\, b. \,\}, P \,\}; \\
C_Q(\{\, a.,\, b. \,\}) &= \{\, \{\, a.,\, b \leftarrow a. \,\}, \{\, a.,\, b. \,\}, P \,\}; \\
C_Q(\{\, b \leftarrow a.,\, b. \,\}) &= \{\, \{\, b. \,\}, \{\, a.,\, b \leftarrow a. \,\}, \{\, a.,\, b. \,\}, \{\, b \leftarrow a., b. \,\}, P \,\}; \\
C_Q(\{\, P \,\}) &= \{\, \{\, a.,\, b \leftarrow a. \,\}, \{\, a.,\, b. \,\}, P \,\}.
\end{aligned}
$$

For the possible entrenchments associated with P, we obtain the following contraction outcomes:

1. If $\{\, a. \,\} \leq_Q \{\, b \leftarrow a. \,\} \leq_Q \{\, b. \,\} \leq_Q \{\, a. \,\}$, then $P \dot{-}_{\leq_Q} Q = \emptyset$.
2. If $\{\, b \leftarrow a. \,\} \leq_Q \{\, b. \,\} \leq_Q \{\, b \leftarrow a. \,\} <_Q \{\, a. \,\}$, then $P \dot{-}_{\leq_Q} Q = \{\, a. \,\}$.
3. If $\{\, a. \,\} \leq_Q \{\, b. \,\} \leq_Q \{\, a. \,\} <_Q \{\, b \leftarrow a. \,\}$, then $P \dot{-}_{\leq_Q} Q = \{\, b \leftarrow a. \,\}$.
4. If $\{\, a. \,\} <_Q \{\, b \leftarrow a. \,\}$ and $\{\, a. \,\} <_Q \{\, b. \,\}$, then $P \dot{-}_{\leq_Q} Q = \{\, b \leftarrow a.,\, b. \,\}$.

■

The operator $\dot{-}_{\leq_Q}$ satisfies all basic AGM contraction postulates except Recovery. This comes at no surprise, given that even FC logic contraction satisfies Recovery only for certain underlying logics.

Theorem 6.36. The contraction operator $\dot{-}_{\leq_Q}$ satisfies ($\dot{-}1$)–($\dot{-}4$) and ($\dot{-}6$).

6.5 Entrenchment Revision

With the assistance of the Levi identity, we can define an entrenchment revision operation as follows.

Definition 6.37 (Entrenchment Revision)**.** Let $P, Q \in \mathcal{LP}_{\mathcal{A}}$ and $\leq_Q$ be an epistemic entrenchment associated with P plus Q. We define an operator $*_{\leq_Q}$ as an *entrenchment revision operator* for P such that

$$P *_{\leq_Q} Q = (P \dot{-}_{\leq_Q} \overline{Q}) + Q.$$

The entrenchment revision operator $*_{\leq_Q}$ satisfies all basic AGM revision postulates.

Theorem 6.38. The revision operator $*_{\leq_Q}$ satisfies ($*1$)–($*6$).

6.6 Discussion

In this chapter, we presented two new approaches to perform belief change in logic programs. Both approaches are built on the concept of ordering the beliefs contained in a logic program by how valuable they are to an agent. Interestingly, even though ensconcement belief change essentially falls under the belief base framework, our ensconcement revision and contraction operators satisfy the majority of AGM postulates. Additionally, our ensconcement contraction performs as well as the original operation in classical logics with respect to the belief base contraction postulates. We also established some useful relationships between our operators. We found that the Levi and Harper identities allow us to properly characterise ensconcement revision and contraction in terms of each other. We also showed that our ensconcement operators are more general than our partial meet operators, and that the choice of granularity of an ensconcement associated with a program does not affect the outcome of an ensconcement revision or contraction operation.

We then defined an entrenchment as an alternative method to order the beliefs expressed by a logic program. In order to construct a contraction operators based on an entrenchment, we had to extend the ordering to include the contracting program and subsequent application of the Levi identity gave rise to an entrenchment revision operator. Our entrenchment operators satisfy only a subset of the AGM postulates. In particular, the satisfiability of the supplementary postulates is hindered by two characteristics of our approach that differ from the classical case. Firstly, in our setting, we are working with logic programs, which are not closed under logical consequence. Secondly, our entrenchment is an ordering only over the initial and the revising/contracting program, not over the entire language. This, in turn, prohibits us to define an entrenchment from a revision or contraction.

A logic program with an associated ensconcement or entrenchment has similar design features as an ordered logic program. An ordered logic program is a tuple $(P, <)$ such that P is a logic program and $<$ is a preference ordering over the rules in P. Thus, it is conceivable to express the revision of P by Q as $(P \cup Q, <)$, where $<$ is some appropriate preference ordering on $P \cup Q$ (from (Delgrande, Schaub & Tompits 2007), for example), and employ one of the different semantics proposed (Brewka & Eiter 1999, Delgrande, Schaub & Tompits 2003, Schaub & Wang 2003) to obtain the preferred answer sets of

this ordered logic program. An ordered logic program can be transformed into a standard logic program, so that the preferred answer sets of the former are exactly the answer sets of the latter (Delgrande et al. 2003). Yet, the transformed program may bear no syntactic relation to the original program. More importantly, an ordered logic program $(P \cup Q, <)$ may be inconsistent, i.e., may have no answer sets, even if P and Q themselves are consistent (Delgrande et al. 2004). These characteristics clearly separate ordered logic programs from our approach here and make them rather unsuitable as a methodology for logic program belief change in general.

7

Connections to Classic Belief Change

The AGM and belief base frameworks for belief change provide elegant structures to define change operations on bodies of beliefs. A crucial part of these frameworks are the postulates that characterise rational behaviour of such change operations. We translated the AGM and belief base postulates to the logic program setting and evaluated each of our operators against them. Tables 7.1–7.4 provide an overview of the postulates that are satisfied by each operator.

Our partial meet revision operator $*_\gamma$ satisfies all basic AGM revision postulates $(*1)$–$(*6)$. However, the partial meet revision operator does not satisfy the supplementary postulates $(*7)$–$(*8)$, even with further restrictions on the selection function, which allow operators under propositional logic to satisfy the supplementary postulates. A similar situation exists for our partial meet contraction operator $\dot{-}_\gamma$. The partial meet contraction operator satisfies all basic postulates $(\dot{-}1)$–$(\dot{-}6)$ with the exception of the controversial Recovery postulate $(\dot{-}5)$. It satisfies $(\dot{-}7)$ but not $(\dot{-}8)$ of the supplementary postulates when the selection function is restricted to be determined by a maximised transitive relation.

On the other hand, evaluating our partial meet revision and contraction op-

erators with respect to the belief base postulates showed that they exhibit the same characteristics as the partial meet base revision and contraction operators for propositional logic (Hansson 1993). The partial meet revision operator is represented by ($*$1b)–($*$5b) (Theorem 5.10) and the partial meet contraction operator by ($\dot{-}$1b)–($\dot{-}$4b) (Theorem 5.18) and thus both operators fit neatly into the belief base framework.

Our ensconcement revision operator $\dot{-}_{\preceq}$ satisfies the same basic AGM postulates ($*$1)–($*$6) as our partial meet revision operator and in addition ($*$8), but also disrespects ($*$7). Our ensconcement contraction operator $\dot{-}_{\preceq}$ satisfies all AGM postulates ($\dot{-}$1)–($\dot{-}$8) with the exception of ($\dot{-}$5). Since an ensconcement construction is essentially geared towards belief bases (Williams 1994), our ensconcement operators should align well with the belief base framework. Indeed, our ensconcement contraction operator $\dot{-}_{\preceq}$ satisfies the same set of belief base postulates as its counterpart for propositional logic (Fermé et al. 2008), that is, ($\dot{-}$1b), ($\dot{-}$2b), and ($\dot{-}$5b)–($\dot{-}$8b). As the classic belief base revision postulates ($\circledast$1)–($\circledast$5) have originally been proposed to characterise partial meet base revision operations, their applicability to characterise ensconcement revision operations is limited. It would be interesting for future work to define a set of belief base revision postulates that can exactly characterise ensconcement revision operations. Until then, we can use our adaptation of the Harper identity to show that any ensconcement contraction operator generated by our ensconcement revision operator $*_{\preceq}$ satisfies ($\dot{-}$1b), ($\dot{-}$2b), and ($\dot{-}$5b)–($\dot{-}$8b).

Despite having just established that our partial meet and ensconcement operators fit properly within the belief base framework, we will nonetheless explain briefly why the AGM postulates ($*$7) and ($\dot{-}$5) pose challenges for our operators. The problem with ($*$7) for our partial meet revision operator $*_{\gamma}$ is that $SE(Q + R) \subseteq SE(Q)$, which means $\mathbb{P}_{Q+R} \subseteq \mathbb{P}_Q$. In the case that $\gamma(\mathbb{P}_{Q+R})$ contains only one element and $\gamma(\mathbb{P}_Q)$ more than one, then applying the intersection over $\gamma(\mathbb{P}_{Q+R})$ and $\gamma(\mathbb{P}_Q)$ may return less rules for $\bigcap \gamma(\mathbb{P}_Q)$ than for $\bigcap \gamma(\mathbb{P}_{Q+R})$, so that $P *_{\gamma} (Q + R) \subseteq (P *_{\gamma} Q) + R$ does not necessarily hold. The situation is similar for our ensconcement revision operator $*_{\preceq}$. Since $SE(Q + R) \subseteq SE(Q)$, we have $SE(cut_{\preceq}(Q + R)) \cap SE(Q + R) \subseteq SE(cut_{\preceq}(Q)) \cap SE(Q)$ whenever $cut_{\preceq}(Q+R) = cut_{\preceq}(Q)$. Thus, it is easier for any rule $r \in P$ to fulfill the condition $SE(cut_{\preceq}(Q+R)) \cap SE(Q+R) \subseteq SE(r)$ than $SE(cut_{\preceq}(Q)) \cap SE(Q) \subseteq SE(r)$ and so $P *_{\preceq} (Q + R) \subseteq (P *_{\preceq} Q) + R$ does not necessarily hold. Regarding postulate ($\dot{-}$5), neither $\dot{-}_{\gamma}$ nor $\dot{-}_{\preceq}$ satisfies ($\dot{-}$5) because the rules contracted from P may differ from the rules in Q,

so that simply adding Q to $P \dot{-}_\gamma Q$ may not restore P. Even if we consider the postulate on a semantic level in the form "$(P \dot{-} Q) + Q \models_s P$", it would not be satisfied by $\dot{-}_\gamma$ or $\dot{-}_\preceq$, because a contracted program $P \dot{-}_\gamma Q$ may share SE models with Q that are not part of the set of SE models of the initial program P.

Our entrenchment revision operator satisfies only the basic AGM revision postulates and our entrenchment contraction operator satisfies only the basic AGM contraction postulates except Recovery. Full compliance is hindered by the circumstances explained in Section 6.6. In fact, our definition of an entrenchment over only the sets of rules that are involved in an entrenchment revision or contraction operation is admittedly not very elegant. As an entrenchment construction is essentially geared towards logically closed sets, we can thus conclude for now that its applicability to logic programs, which are not logically closed, is somewhat limited.

Table 7.1: AGM revision postulates satisfied by $*_{\gamma'}$, $*_{\preceq}$, and $*_{\leq_Q}$

	$(*1)$	$(*2)$	$(*3)$	$(*4)$	$(*5)$	$(*6)$	$(*7)$	$(*8)$
$*_{\gamma'}$	✓	✓	✓	✓	✓	✓		
$*_{\preceq}$	✓	✓	✓	✓	✓	✓		✓
$*_{\leq_Q}$	✓	✓	✓	✓	✓	✓		

Table 7.2: AGM contraction postulates satisfied by $\dot{-}_{\gamma'}$, $\dot{-}_{\preceq}$, and $\dot{-}_{\leq_Q}$

	$(\dot{-}1)$	$(\dot{-}2)$	$(\dot{-}3)$	$(\dot{-}4)$	$(\dot{-}5)$	$(\dot{-}6)$	$(\dot{-}7)$	$(\dot{-}8)$
$\dot{-}_{\gamma'}$	✓	✓	✓	✓		✓	✓	
$\dot{-}_{\preceq}$	✓	✓	✓	✓		✓	✓	✓
$\dot{-}_{\leq_Q}$	✓	✓	✓	✓		✓		

Table 7.3: Base revision postulates satisfied by $*_{\gamma}$ and $*_{\preceq}$

	$(*1b)$	$(*2b)$	$(*3b)$	$(*4b)$	$(*5b)$
$*_{\gamma}$	✓	✓	✓	✓	✓
$*_{\preceq}$	✓	✓			✓

Table 7.4: Base contraction postulates satisfied by $\dot{-}_{\gamma}$ and $\dot{-}_{\preceq}$

	$(\dot{-}1b)$	$(\dot{-}2b)$	$(\dot{-}3b)$	$(\dot{-}4b)$	$(\dot{-}5b)$	$(\dot{-}6b)$	$(\dot{-}7b)$	$(\dot{-}8b)$
$\dot{-}_{\gamma}$	✓	✓	✓	✓	✓	✓		✓
$\dot{-}_{\preceq}$	✓	✓			✓	✓	✓	✓

8

Connections to Logic Program Belief Change

Returning to our motivation, we now give some formal examples to highlight the drawbacks of the distance-based revision operator and then show how our partial meet and ensconcement revision operators addresses these. Recall from Section 3.1 that the distance-based revision operator $\star$ does not specify the structure of the revised program. For convenience, we provide a possible program that corresponds to the revision outcome for each example below, which we denote by $P \star Q$.

1) $P = \{\, a.,\ b \leftarrow a. \,\} \quad Q = \{\, \bot \leftarrow a. \,\}$
$SE(P \star Q) = \{(b, b)\} \quad P \star Q = \{\, \bot \leftarrow a.,\ b. \,\}$

2) $P = \{\, \bot \leftarrow a.,\ b \leftarrow not\, a. \,\} \quad Q = \{\, a. \,\}$
$SE(P \star Q) = \{(ab, ab)\} \quad P \star Q = \{\, a.,\ b. \,\}$

3) $P = \{\, a.,\ b \leftarrow not\, c. \,\} \quad Q = \{\, \bot \leftarrow c. \,\}$
$SE(P \star Q) = \{(ab, ab)\} \quad P \star Q = \{\, a.,\ b.,\ \bot \leftarrow c. \,\}$

4) $P = \{\, a.,\ b \leftarrow not\, a. \,\} \quad Q = \{\, \bot \leftarrow a. \,\}$
$SE(P \star Q) = \{(\emptyset, \emptyset), (\emptyset, b), (b, b)\} \quad P \star Q = \{\, \bot \leftarrow a. \,\}$

5) $P = \{ \bot \leftarrow a.,\ b \leftarrow a. \}$ $\quad Q = \{ a. \}$
$SE(P \star Q) = \{(a, a), (a, ab), (ab, ab)\}$ $\quad P \star Q = \{ a. \}$

In Example 1), the initial belief state expressed by program P consists of a and b. In fact, the second rule in P says that we believe b *if* we believe a. After revising by the program Q, which simply states that we do not believe a, we still believe b even though the reason to believe b is not given anymore. The explanation for this is that the revision operator acts on a program-level, not on a rule-level, as it considers just the SE models of the program in its entirety. However, the dependency of b on a is not captured by the SE models of the program, only by the SE models of the second rule. Therefore, b is treated as an independent fact during the revision process. The situation is similar in Example 2). Here, we initially believe b due to the absence of belief a. After the revision, we continue to believe b even though the grounds for b do not exist anymore.

In Example 3), we are indifferent with respect to b initially and should believe b when we do not believe c. The revising program Q contains information that c indeed does not hold. Thus, b is incorporated into the new belief state. Yet, Example 4) describes a similar scenario in which b is not included in the resulting belief state, thereby showing a clear discrepancy to the behaviour of the revision operator in the previous example. It seems that some dependencies between atoms expressed in P are respected, while others are not.

Comparing Examples 4) and 5) demonstrates another characteristic of the revision operator. In both examples, the revision operation effectively disregards the second rule in P. This is due to the fact that the set of SE models of P is exactly the set of SE models of the first rule. The second rule is thus *locally irrelevant* (Delgrande & Wang 2014). Consequently, the revision operator returns a result as if P had consisted merely of the first rule.

Below are the results of our partial meet revision operator for the five examples above. In each example, the result is independent of the choice of selection function.

1) $SE(P *_\gamma Q) = \{(\emptyset, \emptyset), (\emptyset, b), (b, b)\}$ $\quad P *_\gamma Q = \{ \bot \leftarrow a.,\ b \leftarrow a. \}$

2) $SE(P *_\gamma Q) = \{(a, a), (a, ab), (ab, ab)\}$ $\quad P *_\gamma Q = \{ a.,\ b \leftarrow not\, a. \}$

3) $SE(P *_\gamma Q) = \{(ab, ab)\}$ $\quad P *_\gamma Q = \{ a.,\ b \leftarrow not\, c.,\ \bot \leftarrow c. \}$

4) $SE(P *_\gamma Q) = \{(b, b)\}$ $\quad P *_\gamma Q = \{ \bot \leftarrow a.,\ b \leftarrow not\, a. \}$

5) $SE(P *_\gamma Q) = \{(ab, ab)\}$ $\quad P *_\gamma Q = \{ a.,\ b \leftarrow a. \}$

We can see that partial meet revision addresses the shortcomings of the distance-based revision method. In Examples 1) and 2), the partial meet revision operator preserves the dependency of b on a and $not\, a$, respectively. This is expressed on the syntactic level by the revised program $P *_\gamma Q$ and on the semantic level by $SE(P *_\gamma Q)$. Regarding Examples 3) and 4), the partial meet revision operator treats the dependency of b on $not\, c$ and $not\, a$, respectively, in the same manner and adds b to the belief state uniformly in both examples. Finally, the partial meet revision operator takes into account all rules in a program, even those that may be "invisible" from a purely model-based perspective, as shown by the outcomes for Examples 4) and 5).

We will now examine the behaviour of our ensconcement revision operator. In each example, the revision outcome is independent of a constructed ensconcement associated with P.

1) $SE(P *_\preceq Q) = \{(\emptyset, \emptyset), (\emptyset, b), (b, b)\}$ $\quad P *_\preceq Q = \{ \bot \leftarrow a.,\ b \leftarrow a. \}$

2) $SE(P *_\preceq Q) = \{(a, a), (a, ab), (ab, ab)\}$ $\quad P *_\preceq Q = \{ a.,\ b \leftarrow not\, a. \}$

3) $SE(P *_\preceq Q) = \{(ab, ab)\}$ $\quad P *_\preceq Q = \{ a.,\ b \leftarrow not\, c.,\ \bot \leftarrow c. \}$

4) $SE(P *_\preceq Q) = \{(b, b)\}$ $\quad P *_\preceq Q = \{ \bot \leftarrow a.,\ b \leftarrow not\, a. \}$

5) $SE(P *_\preceq Q) = \{(ab, ab)\}$ $\quad P *_\preceq Q = \{ a.,\ b \leftarrow a. \}$

For all five examples, the ensconcement revision operator $*_\preceq$ returns the same desired results as the partial meet revision operator $*_\gamma$. Examining in particular Examples 4) and 5), it now becomes evident why we diverted in our formulation of Condition ($\preceq$1) from the classic definition. Condition ($\preceq$1) prohibits strict implication on the same ensconcement level. Without this refined requirement, for Example 4) we could construct an ensconcement associated with P such that $a. \preceq b \leftarrow not\, a. \preceq a.$, which would lead to the outcome $P *_\preceq Q = \{ \bot \leftarrow a. \}$. For Example 5), we could construct an ensconcement associated with P such that $\bot \leftarrow a. \preceq b \leftarrow a. \preceq \bot \leftarrow a.$, which would give us the outcome $P *_\preceq Q = \{ a. \}$. These revision outcomes would correspond exactly to the undesired results of the distance-based revision operator $\star$, which

Table 8.1: AGM revision postulates satisfied by $\star$

	$(*1)$	$(*2)$	$(*3)$	$(*4)$	$(*5)$	$(*6)$	$(*7)$	$(*8)$
$\star$	✓				✓	✓		

Table 8.2: Base revision postulates satisfied by $\star$

	$(*1b)$	$(*2b)$	$(*3b)$	$(*4b)$	$(*5b)$
$\star$					✓

we set out to avoid because they disrespect the principle of minimal change on the syntactic level.

It is now left to examine how our operators compare to the distance-based operator in terms of the AGM and belief base postulates. Tables 8.1 and 8.2 display the postulates satisfied by the distance-based operator $\star$. We can see that both our partial meet revision operator $*_\gamma$ and our ensconcement revision operator $*_\preceq$ are better-behaved than $\star$ on the scale of AGM postulates as well as on the scale of belief base postulates. It should be noted, however, that $\star$ satisfies $(*4m)$, which has a stricter antecedent than $(*6)$, but neither $*_\gamma$ nor $*_\preceq$ satisfies $(*4m)$. We demonstrate this in the next two examples.

Example 8.1. Let $P = \{\, a.,\, b.\,\}$, $P' = \{\, a.,\, b \leftarrow a.\,\}$, and $Q = \{\, \bot \leftarrow a.\,\}$. For any selection function γ for P and any selection function γ' for P', we have $\mathbb{P}_Q = \{\,\{\, b.\,\}\,\} = \gamma(\mathbb{P}_Q)$ and $\mathbb{P}'_Q = \{\,\{\, b \leftarrow a.\,\}\,\} = \gamma(\mathbb{P}'_Q)$. Thus, $P *_\gamma Q = \{\, b.,\, \bot \leftarrow a.\,\}$ and $P' *_{\gamma'} Q = \{\, b \leftarrow a.,\, \bot \leftarrow a.\,\}$. ■

Example 8.2. Consider again P, P', and Q from Example 8.1. Let $\preceq$ be the ensconcement associated with P such that $\{\, a.\,\} \prec \{\, b.\,\}$. Then $P *_\preceq Q = \{\, b.,\, \bot \leftarrow a.\,\}$, while for any ensconcement $\preceq'$ associated with P', it holds that $P' *_{\preceq'} Q = \{\, b \leftarrow a.,\, \bot \leftarrow a.\,\}$. ■

Although $P \equiv_s P'$ in these two examples, we find that $P *_\gamma Q = P *_\preceq Q \not\equiv_s P' *_{\preceq'} Q = P' *_{\gamma'} Q$, because $SE(P *_\gamma Q) = SE(P *_\preceq Q) = \{(b, b)\} \neq \{(\emptyset, \emptyset), (\emptyset, b), (b, b)\} = SE(P' *_{\preceq'} Q) = SE(P' *_{\gamma'} Q)$. Indeed, this behaviour is to be somewhat expected. As $(*4m)$ is the postulate that expresses full syntax-independence, it is arguably too strict a requirement for any revision operator that respects semantic as well as syntactic information.

Regarding the screened semi-revision approach by Krümpelmann & Kern-Isberner (2012) (see Section 3.2), we can show that our partial meet revision operator $*_\gamma$ is a generalisation of their screened consolidation operator $!_{\gamma_P}$.

Let $P, Q \in \mathcal{LP}_{\mathcal{A}}$,

$$P \perp_!^{SE} Q = \{\, R \mid Q \subseteq R \subseteq P, SE(R) \neq \emptyset \text{ and for all } R' \text{ with } R \subset R' \subseteq P : SE(R') = \emptyset \,\}$$

and $P !_{\gamma_P}^{SE} Q = \gamma_P(P \perp_!^{SE} Q)$.

Proposition 8.3. Let $P, Q \in \mathcal{LP}_{\mathcal{A}}$. For any maxichoice selection function γ_P for P, there exists a selection function γ for P such that $(P \cup Q) !_{\gamma_P}^{SE} Q = P *_\gamma Q$.

Conversely, if we translate our partial meet revision operator $*_\gamma$ to answer set semantics and restrict the selection function γ to select at most one element, then it will coincide with the screened consolidation operator $!_{\gamma_P}$. Let $P, Q \in \mathcal{LP}_{\mathcal{A}}$,

$$\mathbb{P}_Q^{AS} = \{\, R \subseteq P \mid AS(R \cup Q) \neq \emptyset \text{ and for all } R' \text{ with } R \subset R' \subseteq P : AS(R' \cup Q) = \emptyset \,\},$$

and γ^1 be a selection function that returns exactly one element from a non-empty set, defined formally as follows.

Definition 8.4. A *single-choice selection function* γ^1 for a set S is a function such that:

(1) $\mathbb{S} \subseteq 2^S$,

(2) $\gamma^1(\mathbb{S}) = R$ for some $R \in \mathbb{S}$, and

(3) if $\mathbb{S} \neq \emptyset$, then $\gamma^1(\mathbb{S}) \neq \emptyset$.

Definition 8.5 (Partial Meet Revision under Answer Set Semantics)**.** Let $P \in \mathcal{LP}_{\mathcal{A}}$ and γ^1 be a single-choice selection function for P. A partial meet revision operator $*_{\gamma^1}^{AS}$ for P under answer set semantics is defined such that for any $Q \in \mathcal{LP}_{\mathcal{A}}$:

$$P *_{\gamma^1}^{AS} Q = \begin{cases} P \cup Q & \text{if } AS(Q) = \emptyset \text{ and } \mathbb{P}_Q^{AS} = \emptyset, \\ \gamma^1(\mathbb{P}_Q^{AS}) \cup Q & \text{otherwise.} \end{cases}$$

Proposition 8.6. Let $P, Q \in \mathcal{LP}_{\mathcal{A}}$ and γ^1, γ_P be single-choice and maxichoice selection functions, respectively, for P. If $\gamma^1(2^P) \cup Q = \gamma_P(\{\, R \cup Q \mid R \in 2^P \,\})$ for any γ_P, then $P *_{\gamma^1}^{AS} Q = (P \cup Q) !_{\gamma_P} Q$.

9

Belief Change in Hybrid Knowledge Bases

In the last years, *ontologies* have become widely accepted as knowledge representation formalisms to model concepts and relationships within a domain of expertise. For example, SNOMED CT, the Gene Ontology, and OBO Foundry Ontologies represent standardised biomedical terminologies and are well-established within their field of application. A powerful tool to construct ontologies is the family of languages known as *description logics* (DL) (Baader et al. 2007). DL languages are fragments of first-order logic and enable efficient reasoning over the information specified in an ontology.

More recently, DL languages have been integrated with logic programs in order to combine monotonic with nonmonotonic reasoning (see Section 3.4). For such *hybrid knowledge bases* to be successfully adopted in practical applications, it is crucial that they are not only static entities but can respond to changes as required within the domain. Therefore, robust change mechanisms must be defined. In this chapter, we will focus on exactly this task by investigating whether we can extend the revision and contraction operators for logic programs introduced in the previous chapters to cover belief change in hybrid

knowledge bases that is compliant with the belief base framework.

We first look at *normal DL logic programs* (Shen & Wang 2011), which are expressive hybrid knowledge bases that capture a range of existing hybrid formalisms. A normal DL logic program is a logic program in which DL expressions related to an external DL knowledge base may be included as rule atoms. We present a new semantics for normal DL logic programs, develop a revision operation, and investigate its properties with respect to the belief base revision postulates.

We then focus on a more general formalism, hybrid knowledge bases under $\mathbf{QHT}^s_=$ semantics (Lifschitz, Pearce & Valverde 2007, Pearce & Valverde 2008). This formalism allows us to construct a hybrid knowledge base consisting of two components, a first-order theory and a logic program, with a semantics that combines monotonic reasoning over the theory component with nonmonotonic reasoning over the logic program component (de Bruijn et al. 2010). With no restrictions to a certain fragment of first-order logic or a certain class of logic programs, it is a highly expressive formalism that captures several other approaches to hybrid knowledge bases that were proposed to date.

Two major challenges have impeded the progress on defining belief change operators for hybrid knowledge bases. Firstly, any change operator must be able to handle changes to the theory component, the logic program component, or both, depending on the new information acquired. When either component is changed, it may have an effect on the other component and cause the hybrid knowledge base to become inconsistent. We address this issue by fine-tuning our notion of compatible sets (Definition 5.1) with an input parameter and utilising $\mathbf{QHT}^s_=$ semantics to prevent inconsistencies in the resulting hybrid knowledge base.

The second challenge lies in respecting the monotonic foundation of the description logic or first-order theory component as well as the nonmonotonic foundation of the logic program component. This difference is crucial when dealing with knowledge changes: any conclusion inferred from a theory continues to hold when adding new information, but a conclusion from a logic program may be deleted upon acquisition of new knowledge. Interestingly, $\mathbf{QHT}^s_=$ semantics and our revision and contraction operators for logic programs share a common feature that will help us overcome this challenge. On the one hand, $\mathbf{QHT}^s_=$ integrates the semantics for the theory component and the logic

program component by providing a combined, monotonic characterisation for the entire hybrid knowledge base, built on the *logic of here-and-there* (HT) (Łukasiewicz 1941). On the other hand, SE semantics can be axiomatised via HT logic such that SE models correspond exactly to HT models (Lifschitz et al. 2001, Turner 2003). As such, it turns out that the key for defining belief change operators for logic programs that comply with the belief base framework – the monotonic characterisation via SE/HT models – also paves the way to define rational belief change operators for hybrid knowledge bases.

9.1 Belief Change under Extended Answer Set Semantics

In this section, we focus on belief change in normal DL logic programs. We first develop a three-valued semantics for normal DL logic programs based on Fitting's Kripke-Kleene semantics (Fitting 1985). In contrast to the standard two-valued semantics for normal DL logic programs, the three-valued semantics allows us to propagate assumptions about the non-existence of certain information through a belief change operation. We then give a fixpoint characterisation for normal DL logic programs under the three-valued semantics and use it in the definition of three-valued answer sets. We show that every two-valued model or answer set has a corresponding three-valued model or answer set, respectively. Finally, we use our results to construct revision in normal DL logic programs and show which belief base revision postulates it satisfies.

9.1.1 Three-Valued Answer Sets

We define a three-valued interpretation for normal DL logic programs based on the definition for normal logic programs (Apt & Bol 1994) as follows.

Definition 9.1 (Three-Valued Interpretation)**.** Let Π be a normal DL logic program relative to a DL knowledge base L. A *three-valued interpretation* I of Π relative to L is a pair $\langle I^+, I^- \rangle$ such that I^+ and I^- are subsets of HB_Π and $I^+ \cap I^- = \emptyset$.

The intuition behind three-valued interpretations is that I^+ contains atoms which are assumed to be true and I^- contains atoms which are assumed not

to hold. For any three-valued interpretation I, let $I^+|_\Omega = \{ A \mid A \in I^+$ and $pred(A) \in \Omega \}$ and $\neg I^-|_\Omega = \{ \neg A \mid A \in I^-$ and $pred(A) \in \Omega \}$. These sets are often treated as conjunctions. We call $I = \langle I^+, I^- \rangle$ *consistent with* L if $L \cup I^+|_\Omega \cup \neg I^-|_\Omega$ is consistent. We now adapt the Kripke-Kleene semantics for logic programs (Fitting 1985) to define three-valued models for normal DL logic programs.

Definition 9.2 (Three-Valued Model)**.** Let Π be a normal DL logic program relative to a DL knowledge base L. The *satisfaction* of a literal l under a three-valued interpretation $I = \langle I^+, I^- \rangle$ relative to L, written $I \models_{3,L} l$, is defined as follows:

- A is a pure atom from HB_Π
  - If $A \in I^+$, then $I \models_{3,L} A$;
  - If $A \in I^-$, then $I \models_{3,L} not\, A$.
- A is a ground DL axiom
  - If $L \cup I^+|_\Omega \cup \neg I^-|_\Omega \models A$, then $I \models_{3,L} A$;
  - If $L \cup I^+|_\Omega \cup \neg I^-|_\Omega \models \neg A$, then $I \models_{3,L} not\, A$.
- $I \not\models_{3,L} A$ iff $I \models_{3,L} not\, A$.
- $I \not\models_{3,L} not\, A$ iff $I \models_{3,L} A$.

We say that A is *undefined* iff neither $I \models_{3,L} A$ nor $I \models_{3,L} not\, A$.

Given a rule r in $ground(\Pi)$, we define $I \models_{3,L} body(r)$ iff $I \models_{3,L} l$ for each l in $body(r)$, and $I \not\models_{3,L} body(r)$ otherwise. A rule r is satisfied under I relative to L, written $I \models_{3,L} r$, if $I \not\models_{3,L} body(r)$ or $I \models_{3,L} head(r)$. A three-valued interpretation I is a *three-valued model* of Π relative to L iff:

1. I is consistent with L, and
2. $I \models_{3,L} r$ for all $r \in ground(\Pi)$.

Given two three-valued interpretations $I = \langle I^+, I^- \rangle$ and $E = \langle E^+, E^- \rangle$, we say that E is a *subset* of I, written $E \subseteq I$, iff $E^+ \subseteq I^+$ and $E^- \subseteq I^-$.

While the definition of satisfaction above looks similar to the one for two-valued interpretations (see Section 2.3), three-valued interpretations allow a more fine-grained semantics. Firstly, under the three-valued semantics a negated pure atom $not\, A$ is only satisfied if A is explicitly specified in I^-. Secondly, when the atom A occurring in a literal l is a DL axiom, only the atoms included in $I^-|_\Omega$ are negated (in the sense of strong negation) to check for classical entailment of A or $\neg A$, in order to determine satisfaction. In contrast, considering the two-valued semantics for a two-valued interpretation $\mathcal{I}$, all atoms in $\overline{\mathcal{I}}|_\Omega$ are negated. Therefore, we have a stricter definition of satisfaction under the three-valued semantics. The evaluation of satisfaction relies not only on the atoms in I^+ but also on the atoms in I^-, i.e., the set of atoms for which we explicitly assume that they do not hold. This is demonstrated in the next example.

Example 9.3. Consider the ground program

$$\Pi = \{\, A(a) \leftarrow not\, C(a).,\ \bot \leftarrow B(a).\,\}$$

with $L = \{C \sqsubseteq B\}$, $\Omega = \{B, C\}$, and $HB_\Pi = \{A(a), B(a), C(a)\}$. A two-valued model of Π is $\{A(a)\}$. However, the three-valued interpretation $I = \langle\{A(a)\}, \emptyset\rangle$ is not a three-valued model of Π. Since neither $L \cup I^+|_\Omega \cup \neg I^-|_\Omega = \{C \sqsubseteq B, A(a)\} \not\models B(a)$ nor $L \cup I^+|_\Omega \cup \neg I^-|_\Omega = \{C \sqsubseteq B, A(a)\} \not\models \neg B(a)$, the DL axiom $B(a)$ is undefined and thus the second rule is not satisfied. A three-valued model of Π is $\langle\{A(a)\}, \{B(a)\}\rangle$, in which $B(a)$ is explicitly assumed to be false. ■

We clarify the relationship between two-valued and three-valued models in the following two theorems.

Lemma 9.4. Let Π be a normal DL logic program relative to a DL knowledge base L, l a literal occurring in $ground(\Pi)$, and $I^+ \subseteq HB_\Pi$ consistent with L. Then $I^+ \models_L l$ iff $\langle I^+, \overline{I^+}\rangle \models_{3,L} l$.

Theorem 9.5. Let Π be a normal DL logic program relative to a DL knowledge base L and $I^+ \subseteq HB_\Pi$ consistent with L. Then I^+ is a two-valued model of Π iff $\langle I^+, \overline{I^+}\rangle$ is a three-valued model of Π.

Lemma 9.6. Let Π be a normal DL logic program relative to a DL knowledge base L, l a literal occurring in $ground(\Pi)$, and $\langle I^+, I^-\rangle$ a three-valued interpretation of Π. If I^+ is consistent with L, then $\langle I^+, I^-\rangle \models_{3,L} l$ implies $I^+ \models_L l$.

Theorem 9.7. Let Π be a normal DL logic program relative to a DL knowledge base L and $\langle I^+, I^-\rangle$ a three-valued model of Π. If I^+ is consistent with L, then I^+ is a two-valued model of Π.

We now give a fixpoint characterisation based on the three-valued semantics and define answer sets for normal DL logic programs.

Definition 9.8 (Immediate Consequence Operator)**.** Let Π be a normal DL logic program relative to a DL knowledge base L and let $I = \langle I^+, I^-\rangle$ be a three-valued interpretation of Π that is consistent with L. We define

$$T_\Pi(I^+, I^-) = \{\, A \mid A \leftarrow \mathit{body}(r) \in \mathit{ground}(\Pi) \text{ and } \langle I^+, I^-\rangle \models_{3,L} \mathit{body}(r) \,\},$$

$T^0_\Pi(\emptyset, I^-) = \emptyset$ and $T^{i+1}_\Pi(\emptyset, I^-) = T_\Pi(T^i_\Pi(\emptyset, I^-), I^-)$.

Lemma 9.9. Let Π be a normal DL logic program relative to a DL knowledge base L. For any two three-valued interpretations $\langle J^+_1, I^-\rangle$ and $\langle J^+_2, I^-\rangle$ of Π with $J^+_1 \subseteq J^+_2$, $\langle J^+_1, I^-\rangle \models_{3,L} l$ implies $\langle J^+_2, I^-\rangle \models_{3,L} l$ for each literal l occurring in $\mathit{ground}(\Pi)$.

We show in the next proposition that T_Π is monotonic with regard to its first argument.

Proposition 9.10. Let Π be a normal DL logic program relative to a DL knowledge base L and $I^- \subseteq HB_\Pi$. For any $J^+_1 \subseteq J^+_2 \subseteq HB_\Pi$, $T_\Pi(J^+_1, I^-) \subseteq T_\Pi(J^+_2, I^-)$.

The sequence $\bigcup_{i=0}^{\infty} T^i_\Pi(\emptyset, I^-)$ converges to a fixpoint since it is monotonic by Proposition 9.10 and there is only a finite number of atoms in HB_Π and rules in Π. We denote the fixpoint by $T^\alpha_\Pi(\emptyset, I^-)$ and use it in the following definition of three-valued answer sets.

Definition 9.11 (Three-Valued Answer Set)**.** Let Π be a normal DL logic program relative to a DL knowledge base L and let $I = \langle I^+, I^-\rangle$ be a three-valued model of Π relative to L. We define I as a *three-valued answer set* of Π relative to L iff the following two conditions hold:

1. for every $A \in I^+$, either $A \in T^\alpha_\Pi(\emptyset, I^-)$ or $L \cup T^\alpha_\Pi(\emptyset, I^-)|_\Omega \cup \neg I^-|_\Omega \models A$,
2. for any $J^- \subset I^-$, $\langle I^+, J^-\rangle$ does not satisfy the condition above.

We write $AS_{3,L}(\Pi)$ to denote the set of three-valued answer sets of Π. A normal DL logic program Π relative to a DL knowledge base L is *satisfiable* iff $AS_{3,L}(\Pi) \neq \emptyset$.

The definition ensures that we obtain all possible three-valued answer sets with the "right amount" of assumptions. By Definition 9.8, the choice of I^-, i.e., our assumptions about atoms and ground DL axioms that are not to hold, determines which negative literals are satisfied and which are undefined in the body of a rule r, and thus which $A \in head(r)$ are in $T^{\alpha}_{\Pi}(\emptyset, I^-)$. Consequently, for different I^-, we may have different $T^{\alpha}_{\Pi}(\emptyset, I^-)$ and therefore different candidate answer sets. Condition 2 of Definition 9.11 restricts the candidate answer sets to those where I^- is minimal. The purpose of this is to rule out three-valued answer sets with too many assumptions in I^-. This is illustrated in the next example.

Example 9.12. Let $L = \{A \sqsubseteq \neg C\}$ and

$$\Pi = \{\, A(X) \leftarrow not\, B(X).,\ D(X) \leftarrow A(X), not\, C(X).\,\}$$

with $\mathcal{P}_P = \{B, D\}$, $\mathcal{C} = \{a\}$, and $\Omega = \{A, C\}$. Only $I_0 = \langle \emptyset, \emptyset \rangle$, $I_1 = \langle \{A(a), D(a)\}, \{B(a)\} \rangle$, and $I_2 = \langle \{A(a), D(a)\}, \{B(a), C(a)\} \rangle$ satisfy Condition 1 of Definition 9.11. Since $I_1^- \subseteq I_2^-$, I_2 does not satisfy Condition 2 and thus I_0 and I_1 are the only three-valued answer sets. ■

The following theorem shows when a three-valued answer set corresponds to a two-valued answer set.

Lemma 9.13. Let Π be a normal DL logic program relative to a DL knowledge base L and $\langle I^+, I^- \rangle$ a three-valued interpretation of Π. For any literal l occurring in $ground(\Pi)$ and any $J^+ \subseteq I^+$:

1) $\langle J^+, I^- \rangle \models_{3,L} l$ implies $(J^+, I^+) \models_L l$, and

2) $\langle J^+, \overline{I^+} \rangle \models_{3,L} l$ iff $(J^+, I^+) \models_L l$.

The inverse of 1) in Lemma 9.13, $(J^+, I^+) \models_L l$ implies $\langle J^+, I^- \rangle \models_{3,L} l$, does not hold in general. For instance, let $L = \emptyset$, $J^+ = I^+ = I^- = \emptyset$ and $l = not\, A$ where A is a pure atom. We have that $(J^+, I^+) \models_L l$, but $\langle J^+, I^- \rangle \not\models_{3,L} l$.

Lemma 9.14. Let Π be a normal DL logic program relative to a DL knowledge base L, I^+ a two-valued model of Π, and $\langle I^+, I^-\rangle$ a three-valued model of Π. For every $n \geq 0$, $T^n_\Pi(\emptyset, I^-) \subseteq \mathcal{T}^n_\Pi(\emptyset, I^+)$.

Theorem 9.15. Let Π be a normal DL logic program relative to a DL knowledge base L and $\langle I^+, I^-\rangle$ a three-valued answer set of Π. Then I^+ is a two-valued answer set of Π iff $\langle I^+, \overline{I^+}\rangle$ is a three-valued model of Π.

The next example illustrates why the restriction in Theorem 9.15 is required.

Example 9.16. Let $L = \{A(a)\}$ and $\Pi = \emptyset$ be a normal DL logic program relative to L with $\Omega = \{A\}$. Then $HB_\Pi = \{A(a)\}$, and $\langle\emptyset, \emptyset\rangle$ is a three-valued answer set of Π. But $\emptyset$ is not a two-valued answer set of Π since $L \cup \emptyset|_\Omega \cup \neg\overline{\emptyset}|_\Omega = \{A(a), \neg A(a)\}$ is inconsistent. As a matter of fact, program Π has a unique two-valued answer set $\{A(a)\}$ and another three-valued answer set $\langle\{A(a)\}, \emptyset\rangle$. ■

We can see from the following theorem that every two-valued answer set of a normal DL logic program has a three-valued counterpart.

Lemma 9.17. Let Π be a normal DL logic program relative to a DL knowledge base L, I^+ a two-valued model of Π, and $\langle I^+, \overline{I^+}\rangle$ a three-valued model of Π. For every $n \geq 0$, $T^n_\Pi(\emptyset, \overline{I^+}) = \mathcal{T}^n_\Pi(\emptyset, I^+)$.

Theorem 9.18. Let Π be a normal DL logic program relative to a DL knowledge base L and I^+ a two-valued answer set of Π. Then Π has a three-valued answer set $\langle I^+, I^-\rangle$ for some $I^- \subseteq HB_\Pi$.

It is helpful to have an understanding of strong equivalence for normal DL logic programs under three-valued answer set semantics, which can be formalised as follows.

Definition 9.19 (Strong Equivalence of Normal DL Logic Programs)**.** Let Π_1, Π_2 be normal DL logic programs relative to a DL knowledge base L. We say that Π_1 and Π_2 are *strongly equivalent*, denoted $\Pi_1 \equiv_{3,s} \Pi_2$, iff $AS_{3,L}(\Pi_1 \cup \Pi_3) = AS_{3,L}(\Pi_2 \cup \Pi_3)$ for any normal DL logic program Π_3 relative to L.

9.1.2 Revision in Normal DL Logic Programs

Having established a three-valued answer semantics for normal DL logic programs, we can now turn our attention to revising normal DL logic programs.

We base the following definitions of a canonical normal DL logic program and compatible set of a normal DL logic program on the ideas from the program-level approach to logic program revision (Delgrande 2010).

Definition 9.20 (Canonical Normal DL Logic Program)**.** Let Π be a normal DL logic program relative to a DL knowledge base L. For any $I = \langle I^+, I^- \rangle \in AS_{3,L}(\Pi)$, the *canonical normal DL logic program relative to L for I* is

$$\Pi_{can}(I) = \{\, A \leftarrow \mid\ A \in I^+ \,\} \cup \{\, \leftarrow B \ \mid\ B \in I^- \,\}.$$

Definition 9.21 (Compatible Set of a Normal DL Logic Program)**.** Let Π_1, Π_2 be normal DL logic programs relative to a DL knowledge base L. The set of *compatible sets of* Π_1 *with respect to* Π_2 is

$$\begin{aligned}\Pi_1 \downarrow \Pi_2 = \{\, \pi \cup \Pi_{can}(I_2) \mid\ & \pi \subseteq \Pi_1, I_2 = \langle I_2^+, I_2^- \rangle \in AS_{3,L}(\Pi_2), \text{ and} \\ & \exists I = \langle I^+, I^- \rangle \in AS_{3,L}(\pi \cup \Pi_{can}(I_2) \cup \Pi_2) \\ & \text{such that } I_2^- \subseteq I^-, \\ & \text{and for all } \pi' \text{ with } \pi \subset \pi' \subseteq \Pi_1 : \\ & \nexists J = \langle J^+, J^- \rangle \in AS_{3,L}(\pi' \cup \Pi_{can}(I_2) \cup \Pi_2) \\ & \text{such that } I_2^- \subseteq J^- \}.\end{aligned}$$

We need to include Π_2 besides $\Pi_{can}(I_2)$ for any compatible set because Π_2 may include rules that were not used in the construction of the answer set I_2. The condition $I_2^- \subseteq I^-$ is necessary to propagate the set of assumptions in I_2 to I. To define a rule revision operator for normal DL logic programs, we use a single-choice selection function as given in Definition 8.4.

Definition 9.22 (Rule Revision in Normal DL Logic Programs)**.** Let Π_1 be a normal DL logic program relative to a DL knowledge base L and γ^1 a single-choice selection function for Π_1. A *rule revision operator* $*^n_{\gamma^1}$ for Π_1 is defined such that for any normal DL logic program Π_2 relative to L:

$$\Pi_1 *^n_{\gamma^1} \Pi_2 = \begin{cases} \Pi_1 +^h \Pi_2 & \text{if } \Pi_2 \text{ is not satisfiable,} \\ \gamma^1(\Pi_1 \downarrow \Pi_2) +^h \Pi_2 & \text{otherwise.} \end{cases}$$

The definition of $*^n_{\gamma^1}$ is similar to the one of our partial meet revision operator $*_\gamma$, with the exception that we choose exactly one element of the set of compatible sets. We require this restriction to obtain a satisfiable outcome,

whenever the revising program itself is satisfiable, since the underlying semantics is nonmonotonic. We illustrate the revision operation in the next example.

Example 9.23. Consider the DL knowledge base $L = \{ B \sqsubseteq C \}$ and the two programs relative to L

$$\Pi_1 = \{ D(X) \leftarrow not\, C(X)., B(a). \},$$
$$\Pi_2 = \{ A(X) \leftarrow not\, C(X). \}$$

over a shared vocabulary with $\mathcal{P}_P = \{A, D\}$, $\mathcal{C} = \{a\}$, and $\Omega = \{B, C\}$. We have $HB_{\Pi_1} = HB_{\Pi_2} = \{A(a), B(a), C(a), D(a)\}$, $ground(\Pi_1) = \{ D(a) \leftarrow not\, C(a)., B(a). \}$, and $ground(\Pi_2) = \{ A(a) \leftarrow not\, C(a) \}$. The only three-valued answer set for Π_2 is $I_2 = \langle \{A(a)\}, \{C(a)\} \rangle$. Thus, $\Pi_{can}(I_2) = \{ A(a)., \bot \leftarrow C(a). \}$ and

$$\Pi_1 \downarrow \Pi_2 = \{\{ D(X) \leftarrow not\, C(X)., A(a)., \bot \leftarrow C(a). \}\}.$$

We obtain

$$\Pi_1 *^n_{\gamma^1} \Pi_2 = \{ D(X) \leftarrow not\, C(X)., A(a)., \bot \leftarrow C(a)., A(X) \leftarrow not\, C(X). \}.$$

We have to eliminate the rule $B(a)$. due to the assumption that $C(a)$ does not hold and the inclusion axiom in L. The only three-valued answer set for $\Pi_1 *^n_{\gamma^1} \Pi_2$ is $\langle \{A(a), D(a)\}, \{C(a)\} \rangle$. ■

The example above also shows why a three-valued semantics is needed in the definition of our revision. In the two-valued case, we would have the answer set $\{A(a)\}$ for Π_2. Then, we would not have to eliminate $B(a)$, and $\{A(a), B(a), C(a)\}$ would be the answer set of the revision sequence. Yet, the reason for $A(a)$ to hold in Π_2 should be preserved, i.e., $C(a)$ should not appear in the answer set. Note that facts may be eliminated during a revision in order to preserve the justification of answer sets from the revising program, as pointed out by Delgrande (2010). Facts can be protected by re-stating them in the revising program.

The following theorem specifies which belief base postulates are satisfied by the revision operator.

Theorem 9.24. The revision operator $*^n_{\gamma^1}$ satisfies the postulates ($*^h$1b) and ($*^h$5b).

Postulate ($*^h$2b) is not fulfilled because $*^n_{\gamma^1}$ returns also rules from a canonical program $\Pi_{can}(I_2)$, which may not be in Π_1 or Π_2. ($*^h$3b) and ($*^h$4b) are not fulfilled because the construction of compatible sets of a normal DL logic program relies on satisfiability of some subset of Π_1 not with Π_2 but on satisfiability with Π_2 and a canonical program for some answer set of Π_2.

9.2 Belief Change under Extended HT Semantics

Belief change is about altering a knowledge base by eliminating some information from it and/or adding some information to it. Since hybrid knowledge bases consist of two components, we should distinguish which component is altered during a change operation. The following definition of subsets of hybrid knowledge bases will help us to do so.

Definition 9.25 (Subsets of Hybrid Knowledge Bases)**.** For any two pairs of sets $K = (T, P)$, $K' = (T', P')$,

$$
\begin{aligned}
&K \subseteq K' \text{ iff } T \subseteq T' \text{ and } P \subseteq P',\\
&K \subset K' \text{ iff } K \subseteq K' \text{ and } K' \not\subseteq K,\\
&K \subseteq_T K' \text{ iff } T \subseteq T', \text{ and if } T = T' \text{ then } P \subseteq P',\\
&K \subset_T K' \text{ iff } K \subseteq_T K' \text{ and } K' \not\subseteq_T K,\\
&K \subseteq_P K' \text{ iff } P \subseteq P', \text{ and if } P = P' \text{ then } T \subseteq T',\\
&K \subset_P K' \text{ iff } K \subseteq_P K' \text{ and } K' \not\subseteq_P K.
\end{aligned}
$$

9.2.1 Revision in Hybrid Knowledge Bases

From the previous chapters we have two different constructions for logic program revision at our disposal, which we could extend to hybrid knowledge bases. Upon closer inspection however, it becomes clear that the constructions based on an ensconcement or entrenchment are not really suitable for hybrid knowledge bases. Since hybrid knowledge bases consist of two components, the nature of information should be taken into account by any practical revision operator. For instance, if the revising hybrid knowledge base has an empty logic program component, then it may be desired to preserve the logic program component of the original hybrid knowledge base as much as possible. And if the revising hybrid knowledge base has an empty theory

component, then it may be desired to preserve the theory component of the original hybrid knowledge base as much as possible. We can achieve this in an adaptation of the partial meet construction by fine-tuning the definition of compatible sets. Conversely, an ensconcement or entrenchment revision relies on an ordering that could be defined either over the entire hybrid knowledge base, which would not allow a separate treatment for either component, or over each individual component, which would neglect the interaction between the two components and possible inconsistencies. Consequently, we focus on a partial meet construction and draw on our results for logic programs.

Formally, we fine-tune the notion of compatible sets (Definition 5.1) as follows.

Definition 9.26 (Compatible Set of a HKB)**.** Let $K, K' \in \mathcal{K}_\mathcal{L}$. The set of *compatible sets of K with respect to K'* is

$$\begin{aligned}\mathbb{K}_{K'} = \{\, \kappa = (\tau, \rho) \mid\ & \kappa \subseteq K,\, QHT(\kappa) \cap QHT(K') \neq \emptyset, \\ & \text{and for all } \kappa' = (\tau', \rho') \text{ with } \kappa \subset_\circ \kappa' \subseteq K : \\ & QHT(\kappa') \cap QHT(K') = \emptyset \,\},\end{aligned}$$

where $\subset_\circ = \subset_P$ if $K' = (T', \emptyset)$, $\subset_\circ = \subset_T$ if $K' = (\emptyset, P')$, and $\subset_\circ = \subset$ if $K' = (T', P')$.

Obviously, the inputs for the parameter $\subset_\circ$ can easily be defined differently, for example, if the theory component has absolute priority over the program component, then $\subset_\circ$ could be fixed to $\subset_T$. We now define a partial meet revision operator for hybrid knowledge bases. In the following, let $\bigcap\{(T_1, P_1), \ldots, (T_n, P_n)\} = (\bigcap\{T_1, \ldots, T_n\}, \bigcap\{P_1, \ldots, P_n\})$ for any pairs of sets $(T_1, P_1), \ldots, (T_n, P_n)$.

Definition 9.27 (Partial Meet Revision in HKBs)**.** Let $K \in \mathcal{K}_\mathcal{L}$ and γ be a selection function for K. A *partial meet revision operator* $*^h_\gamma$ for K is defined such that for any $K' \in \mathcal{K}_\mathcal{L}$:

$$K *^h_\gamma K' = \begin{cases} K +^h K' & \text{if } K' \text{ is not satisfiable,} \\ \bigcap \gamma(\mathbb{K}_{K'}) +^h K' & \text{otherwise.} \end{cases}$$

We obtain the following representation theorem with respect to the postulates ($*^h$1b)–($*^h$5b), which corresponds to our representation theorem for partial meet revision in logic programs.

Lemma 9.28. Let $K, K' \in \mathcal{K}_\mathcal{L}$ and γ be a selection function for K. If $\kappa \subseteq K \cap K'$, then $\kappa \subseteq \bigcap \gamma(\mathbb{K}_{K'})$.

Theorem 9.29. Let $K \in \mathcal{K}_\mathcal{L}$. An operator $*^h_\gamma$ is a partial meet revision operator for K generated by some selection function γ for K iff $*^h_\gamma$ satisfies ($*^h$1b)–($*^h$5b).

9.2.2 Contraction in Hybrid Knowledge Bases

As discussed above for revision, ensconcement or entrenchment constructions do not offer as much flexibility as a partial meet construction for hybrid knowledge bases, so that we only focus on the latter here to define a partial meet contraction operator as follows. We first introduce notation for all subsets of K that do not imply K':

$$\begin{aligned}\mathbb{K}^-_{K'} = \{\, \kappa = (\tau, \rho) \mid\ & \kappa \subseteq K,\, QHT(\kappa) \cap \overline{QHT(K')} \neq \emptyset, \\ & \text{and for all } \kappa' = (\tau', \rho') \text{ with } \kappa \subset_\circ \kappa' \subseteq K : \\ & QHT(\kappa') \cap \overline{QHT(K')} = \emptyset \,\},\end{aligned}$$

where $\subset_\circ = \subset_P$ if $K' = (T', \emptyset)$, $\subset_\circ = \subset_T$ if $K' = (\emptyset, P')$, and $\subset_\circ = \subset$ if $K' = (T', P')$.

Analogously to Definition 9.26, the parameter $\subset_\circ$ allows us to prioritise either component of a hybrid knowledge base during a contraction operation, defined as follows.

Definition 9.30 (Partial Meet Contraction in HKBs)**.** Let $K \in \mathcal{K}_\mathcal{L}$ and γ be a selection function for K. A *partial meet contraction operator* $\dot{-}^h_\gamma$ for K is defined such that for any $K' \in \mathcal{K}_\mathcal{L}$:

$$K \dot{-}^h_\gamma K' = \begin{cases} K & \text{if } \not\models K', \\ \bigcap \gamma(\mathbb{K}^-_{K'}) & \text{otherwise.} \end{cases}$$

The following representation theorem holds with respect to the postulates ($\dot{-}^h$1b)–($\dot{-}^h$4b), which corresponds to our representation theorem for partial meet contraction in logic programs.

Theorem 9.31. Let $K \in \mathcal{K}_\mathcal{L}$. An operator $\dot{-}^h_\gamma$ is a partial meet contraction operator for K generated by some selection function γ for K iff $\dot{-}^h_\gamma$ satisfies ($\dot{-}^h$1b)–($\dot{-}^h$4b).

9.3 Discussion

In this chapter, we extended belief change constructions for logic programs to knowledge bases that combine a logic program with a first-order theory. First, we looked at normal DL logic programs under extended answer set semantics and provided a revision operator. Next, we considered hybrid knowledge bases under an alternative semantics, namely $\mathbf{QHT}^s_=$ semantics, a general formalism that has no restrictions on the type of logic program or theory to be combined, and extended our partial meet revision and contraction operators from Chapter 5 to cover belief change.

Comparing the two methods, we obtain the following findings. The extension to hybrid knowledge bases under $\mathbf{QHT}^s_=$ semantics was straightforward and preserved all properties with respect to the belief base revision and contraction postulates. In addition, we were able to fine-tune a change operation so that the outcome is dependent on the nature of information in a hybrid knowledge base, that is, whether information is specified in the logic program or the theory component.

On the other hand, defining belief change under extended answer set semantics was quite cumbersome and gave less impressive results. We first defined a three-valued answer set semantics for normal DL logic programs to allow the propagation of assumptions through a change operation. We then defined a revision operator to change the rule component of a normal DL logic program. Inspection of the belief base revision postulates showed that the revision operator only satisfies the postulates ($*^h$1b) and ($*^h$5b). Besides this, the revision operator has several further limitations. The revision operator assumes a static DL knowledge base, which relates to revision of a hybrid knowledge base under $\mathbf{QHT}^s_=$ semantics where the revising hybrid knowledge base has an empty theory component. Further research is required to address changes on the DL knowledge base level in normal DL logic program revision. Additionally, it is unclear how to define a contraction operation for hybrid knowledge bases based on extended answer set semantics. The complement of a set of answer sets is not adequate for use in the construction of compatible sets for the purpose of defining contraction, due to the nonmonotonicity of the semantics.

In summary, it was instructive to compare two different forms of hybrid knowledge bases with respect to defining belief change. From the findings above and based on the fact that hybrid knowledge bases under $\mathbf{QHT}^s_=$ se-

mantics are strictly more expressive than normal DL logic programs under extended answer set semantics, the former may be a more desirable framework to cover belief change in hybrid knowledge bases.

10 Conclusion

In this thesis, we proposed several operators for belief change in logic programs and hybrid knowledge bases. Our goal was to respect the information expressed by a program on both the program-level and the rule-level during a change operation. In addition, we were guided by the AGM and belief base postulates for revision and contraction and focussed on efficiency to promote deployment in practical applications. We achieved our goals by adapting classic belief change constructions and using the SE semantics for logic programs to define revision and contraction operators for logic programs with natural extensions to hybrid knowledge bases. In particular, our choice of constructions facilitated the definition of syntax-preserving operators that bridge the gap between semantic and syntactic approaches. We summarise here the contributions of our study and highlight some directions for future research.

10.1 Summary of Contributions

We first translated the classic revision and contraction postulates from the AGM and belief base frameworks to logic programs and hybrid knowledge bases. We discussed their inter-relationships and how they relate to previous

translations. It turned out that our translation is closer to the original formulation than previous translations. We also adapted the Levi and Harper identities to the logic program setting.

We then adapted the idea of a partial meet construction to introduce syntax-preserving revision and contraction operators for logic programs. This approach not only allows the operators to preserve more information from a logic program than pure semantic operators, but it also facilitated a natural definition of a contraction operator for logic programs, the first in the field to the best of our knowledge. For both partial meet revision and contraction, we provided a representation theorem that places them firmly within the belief base framework. We used the Levi and Harper identities to characterise partial meet revision and contraction in terms of each other. We further introduced the notion of a module as a subset of a program and defined which modules are the relevant ones with respect to a revision or contraction operation. We then provided an optimisation algorithm for revision and contraction, which reduces the revision or contraction of a program to revising or contracting only the relevant modules of that program.

To explicitly take into account that some beliefs in a knowledge base may be more important than others, we adapted two additional constructions from classic belief change that are based on an ordering over the beliefs represented by a logic program, namely, ensconcement belief change and entrenchment belief change. We began with establishing an ensconcement associated with a logic program as an ordering over the rules contained in the program. Our definition of an ensconcement differs significantly from the original definition for propositional logic, to ensure that the revision and contraction operators we defined via ensconcements deliver intuitive results. Despite this adjustment, our ensconcement operators perform as well as the classic operators with respect to the belief base postulates. As before for our partial meet operators, we applied the Levi and Harper identities to characterise ensconcement revision and contraction in terms of each other. We further showed that our ensconcement operators are more general than our partial meet operators and found that it does not affect the result of an ensconcement revision or contraction whether the ensconcement is constructed over rules or subsets of a program.

Next, we introduced the notion of an entrenchment associated with a logic program. In our definition of an entrenchment, an ordering is created only over the subsets of the programs involved in a change operation, not over the

whole language. This is a particularly useful modification from the original definition to reduce the costs of constructing an entrenchment. We presented an entrenchment contraction operator and used the Levi identity to define an entrenchment revision operator. We showed that our entrenchment revision operator satisfies all basic AGM revision postulates and our entrenchment contraction operator all basic AGM contraction postulates except Recovery.

To conclude our proposal of belief change operators for logic programs, we compared our approach to two state-of-the-art logic program revision methods. We demonstrated, on the one hand, that our partial meet and ensconcement revision operators align more closely to the original AGM and belief base frameworks than the distance-based revision approach (Delgrande, Schaub, Tompits & Woltran 2013), and, on the other hand, that they remedy the unintuitive behaviour of the latter. In addition, we showed that our partial meet revision operator generalises the screened consolidation operator from the screened semi-revision approach (Krümpelmann & Kern-Isberner 2012), and since our ensconcement revision operator generalises our partial meet revision operator, this result also holds for our ensconcement revision operator.

Revisiting our two motivating real-life examples from the introduction, we can see that both our partial meet revision operator as well as our ensconcement revision operator exhibit the desired behaviour. The first example ("fireworks") was formalised as Example 5) in our set of five running examples (see Chapter 8), where $a = fog$ and $b = no\_fireworks$. Applying either of our revision operators to this scenario leaves us with the beliefs that it will be foggy and that there are no fireworks whenever it is foggy, which, under normal deduction, leads us also to believe that the fireworks will be cancelled. This example further demonstrates that our operators do not *harmfully eliminate* (Inoue & Sakama 2004) rules as do revision operators that consider only the set of SE models of a program. The second example ("101 or 280") was formalised as Example 1), where $a = 101\_roadworks$ and $b = 280\_quicker$. Applying either of our revision operators to this example leaves us believing that there are no roadworks on the 101 and that the 280 is quicker whenever there are roadworks on the 101, but not that the 280 is still the better choice. This example also serves to validate that our operators respect the property of *support* (Slota & Leite 2014) that is lacking in revision approaches that operate purely on the program-level.

Finally, our work on logic programs motivated us to investigate how we

could lift our results to belief change in hybrid knowledge bases. With a number of hybrid formalisms available, we chose two of the most expressive ones for our investigations, hybrid knowledge bases under $\mathbf{QHT}^s_=$ semantics and normal DL logic programs under extended answer set semantics. Since $\mathbf{QHT}^s_=$ semantics is an extension of SE semantics, we could adapt our partial meet revision and contraction operators for logic programs in a straightforward fashion to hybrid knowledge bases under $\mathbf{QHT}^s_=$ semantics and fine-tune them to differentiate between the program and theory components as desired. We delivered representation theorems for partial meet revision and contraction in hybrid knowledge bases with respect to the extended belief base postulates. For normal DL logic programs, the definition of belief change was more cumbersome. We first introduced a three-valued answer set semantics for normal DL logic programs and then defined a logic program revision operator for normal DL logic programs. Comparing the two revision operations, it became clear that the former is not only the more general but also the more elegant solution.

10.2 Future Directions

Although our ensconcement and entrenchment belief change operators specify how a program changes during a revision or contraction operation, they do not specify how the associated ensconcement or entrenchment relation changes. An ensconcement or entrenchment is associated with a program before the change operation. During the change operation, some rules may be discarded from the program and some new rules may be added to it in the case of revision, and any effects on the initial ensconcement or entrenchment should be taken into consideration. This problem of changing the ordering over beliefs as part of a belief change operation has been covered for classical logics (Hansson 1995, Nayak, Nelson & Polansky 1996) and later in the context of updates to dynamic prioritised logic programs (Alferes & Pereira 2000). A dynamic prioritised program consists of a sequence `P` of logic programs, which represent some knowledge of an agent, and a sequence `R` of logic programs, which encode a preference relation on the rules of programs in `P`. The update operation is performed in two steps: first, some rules are rejected through a syntactic transformation to obtain a consistent set of rules, then, preferences are applied to find the preferred answer sets. Interestingly, since preferences are modelled in the form of logic programs, it is possible to update the preference relation itself. As part of

our future work, we will consider changes to an ensconcement or entrenchment relation in the context of logic programs revisions and contractions and relate to existing approaches in this field.

Forgetting (Lin & Reiter 1994, Lang, Liberatore & Marquis 2003) is an operation on knowledge bases to discard information by eliminating certain literals from the knowledge base. It has been applied to logic programs in the form of both syntactic and semantic techniques, which are able to characterise logic program updates (Zhang & Foo 2006, Eiter & Wang 2008). Forgetting has further been applied to HEX-Programs (Eiter, Ianni, Schindlauer, Tompits & Wang 2006), which are loose integrations of logic programs and ontologies into hybrid formalisms under generalised answer set semantics (Eiter et al. 2005). More recently, the notion of forgetting has been defined for HT logics (Wang, Zhang, Zhou & Zhang 2014) and logic programs under SE semantics (Delgrande & Wang 2015). These results and developments encourage a study on the relationship between forgetting in logic programs and our frameworks for belief change in logic programs presented here.

While we have provided elegant mechanisms for revising and contracting hybrid knowledge bases under $\mathbf{QHT}^s_=$ semantics, evaluating other formalisms that integrate logic programs and ontologies may lead to more efficient constructions. Specifically, since $\mathbf{QHT}^s_=$ semantics are an extension of HT semantics, under which satisfiability problems are NP-complete (Pearce, Tompits & Woltran 2009), revision and contraction operations on hybrid knowledge bases under $\mathbf{QHT}^s_=$ semantics fall outside of polynomial complexity. It will be useful to investigate further the recent developments in the field of ontology evolution, which led to efficient belief change operations on description logics. In general, description logic change operators can be classified along three dimensions: a) whether they are model-based or syntax-based, b) whether they allow changes to the ABox, TBox, or both, and c) which description logic language they operate on. De Giacomo, Lenzerini, Poggi & Rosati (2009) presented a model-based semantics for ABox changes in the DL-Lite family while keeping axioms in the TBox static. Liu, Lutz, Miličić & Wolter (2011) extended the investigation of ABox changes to more expressive description logics and proposed four alternative model-based change semantics. In contrast, the work by Qi & Du (2009) considered TBox changes. They introduced three model-based change operators and referred to the concept of forgetting in DL-Lite (Wang, Wang, Topor & Pan 2010) to compute the desired result. However, model-based approaches have two drawbacks: the outcome of a change opera-

tion in a DL-Lite knowledge base may not be expressible in the same language (Calvanese, Kharlamov, Nutt & Zheleznyakov 2010*a*, Wang, Wang, Topor & Pan 2008), and, more generally, a description logic knowledge base may have an infinite number of models, which makes a model-based change operation difficult to define and implement (Wang, Wang & Topor 2010*b*). Therefore, Calvanese, Kharlamov, Nutt & Zheleznyakov (2010*b*) presented two syntax-based approaches for ABox changes, and Zheleznyakov, Calvanese, Kharlamov & Nutt (2010) one for TBox changes. De Giacomo et al. (2009) solved the problem of non-expressibility in DL-Lite by computing only the best approximation of a change operation, which is still expressible in the same language. A different approach to change DL-Lite knowlege bases was taken by Wang, Wang & Topor (2010*a*, 2015). On the one hand, they considered changes to the knowledge base in its entirety, i.e., to both the ABox and TBox. On the other hand, their approach is neither model-based nor syntax-based. Instead, they defined the language semantics through so-called *features* and offered two change operators. For the specific DL-Lite language used in their work, the use of features has the benefit that a knowledge base can have only finitely many features. Building on these results, polynomial time algorithms have been presented for ABox changes (Kharlamov, Zheleznyakov & Calvanese 2013) and TBox changes (Zhuang, Wang, Wang & Qi 2014) specified in fragments of DL-Lite. These approaches will provide the basis for our future research on optimising a construction for belief change in hybrid knowledge bases.

References

Alchourrón, C. E., Gärdenfors, P. & Makinson, D. (1985), 'On the logic of theory change: Partial meet contraction and revision functions', *Journal of Symbolic Logic* **50**(2), 510–530.

Alferes, J. J., Banti, F., Brogi, A. & Leite, J. (2005), 'The refined extension principle for semantics of dynamic logic programming', *Studia Logica* **79**(1), 7–32.

Alferes, J. J. & Pereira, L. M. (2000), Updates plus preferences, *in* 'Logics in Artificial Intelligence', Vol. 1919 of *Lecture Notes in Computer Science*, pp. 345–360.

Apt, K. R. & Bol, R. N. (1994), 'Logic programming and negation: A survey', *The Journal of Logic Programming* **19–20**(Supplement 1), 9–71.

Artale, A., Calvanese, D., Kontchakov, R. & Zakharyaschev, M. (2009), 'The DL-Lite family and relations', *Journal of Artificial Intelligence Research* **36**(1), 1–69.

Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D. & Patel-Schneider, P. F., eds (2007), *The Description Logic Handbook: Theory, Implementation, and Applications*, second edn, Cambridge University Press.

Baader, F. & Nutt, W. (2007), *Basic description logics*, in Baader et al. (2007), second edn, chapter 2, pp. 47–104.

Baral, C. (2003), *Knowledge Representation, Reasoning and Declarative Problem Solving*, Cambridge University Press.

Binnewies, S., Wang, Y., Stantic, B. & Wang, K. (2013), Rule revision in Normal DL Logic Programs, *in* 'Web Reasoning and Rule Systems', Vol. 7994 of *Lecture Notes in Computer Science*, pp. 204–209.

Binnewies, S., Zhuang, Z. & Wang, K. (2015), Partial meet revision and contraction in logic programs, *in* 'Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, AAAI 2015', pp. 1439–1445.

Brewka, G. & Eiter, T. (1999), 'Preferred answer sets for extended logic programs', *Artificial Intelligence* **109**(1–2), 297–356.

Brewka, G., Eiter, T. & Truszczyński, M. (2011), 'Answer set programming at a glance', *Communications of the ACM* **54**(12), 92–103.

Brooks, D. R., Erdem, E., Minett, J. W. & Ringe, D. (2005), Character-based cladistics and answer set programming, *in* 'Practical Aspects of Declarative Languages', Vol. 3350 of *Lecture Notes in Computer Science*, pp. 37–51.

Bruijn, J. D., Eiter, T., Polleres, A. & Tompits, H. (2011), 'Embedding non-ground logic programs into autoepistemic logic for knowledge-base combination', *ACM Transactions on Computational Logic* **12**(3), 20:1–20:39.

Calvanese, D., Kharlamov, E., Nutt, W. & Zheleznyakov, D. (2010*a*), Evolution of DL-Lite knowledge bases, *in* 'The Semantic Web – ISWC 2010', Vol. 6496 of *Lecture Notes in Computer Science*, pp. 112–128.

Calvanese, D., Kharlamov, E., Nutt, W. & Zheleznyakov, D. (2010*b*), Updating ABoxes in DL-Lite, *in* 'Proceedings of the 4th Alberto Mendelzon International Workshop on Foundations of Data Management', Vol. 619 of *CEUR Workshop Proceedings*.

Colmerauer, A. & Roussel, P. (1996), The birth of Prolog, *in* 'History of Programming languages–II', pp. 331–367.

Dalal, M. (1988), Investigations into a theory of knowledge base revision: Preliminary report, *in* 'Proceedings of the Seventh AAAI National Conference on Artificial Intelligence, AAAI 1988', pp. 475–479.

de Bruijn, J., Pearce, D., Polleres, A. & Valverde, A. (2010), 'A semantical framework for hybrid knowledge bases', *Knowledge and Information Systems* **25**(1), 81–104.

de Giacomo, G., Lenzerini, M., Poggi, A. & Rosati, R. (2009), 'On instance-level update and erasure in description logic ontologies', *Journal of Logic and Computation* **19**(5), 745–770.

Delgrande, J. P. (2010), 'A program-level approach to revising logic programs under the answer set semantics', *Theory and Practice of Logic Programming* **10**(Special Issue 4–6), 565–580.

Delgrande, J. P., Peppas, P. & Woltran, S. (2013), AGM-style belief revision of logic programs under answer set semantics, *in* 'Logic Programming and Nonmonotonic Reasoning', Vol. 8148 of *Lecture Notes in Computer Science*, pp. 264–276.

Delgrande, J. P., Schaub, T. & Tompits, H. (2003), 'A framework for compiling preferences in logic programs', *Theory and Practice of Logic Programming* **3**(2), 129–187.

Delgrande, J. P., Schaub, T. & Tompits, H. (2007), A preference-based framework for updating logic programs, *in* 'Logic Programming and Nonmonotonic Reasoning', Vol. 4483 of *Lecture Notes in Computer Science*, pp. 71–83.

Delgrande, J. P., Schaub, T., Tompits, H. & Wang, K. (2004), 'A classification and survey of preference handling approaches in nonmonotonic reasoning', *Computational Intelligence* **20**(2), 308–334.

Delgrande, J. P., Schaub, T., Tompits, H. & Woltran, S. (2013), 'A model-theoretic approach to belief change in answer set programming', *ACM Transactions on Computational Logic* **14**(2), 14:1–14:46.

Delgrande, J. P. & Wang, K. (2014), An approach to forgetting in disjunctive logic programs that preserves strong equivalence, *in* 'Proceedings of the Fifteenth International Workshop on Non-Monotonic Reasoning, NMR 2014', pp. 38–44.

Delgrande, J. P. & Wang, K. (2015), A syntax-independent approach to forgetting in disjunctive logic programs, *in* 'Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, AAAI 2015', pp. 1482–1488.

Doyle, J. (1979), 'A truth maintenance system', *Artificial Intelligence* **12**(3), 231–272.

Drabent, W., Eiter, T., Ianni, G., Krennwallner, T., Lukasiewicz, T. & Małuszyński, J. (2009), Hybrid reasoning with rules and ontologies, *in* 'Semantic Techniques for the Web', Vol. 5500 of *Lecture Notes in Computer Science*, pp. 1–49.

Drabent, W. & Małuszyński, J. (2007), Well-founded semantics for hybrid rules, *in* 'Web Reasoning and Rule Systems', Vol. 4524 of *Lecture Notes in Computer Science*, pp. 1–15.

Eiter, T., Fink, M., Sabbatini, G. & Tompits, H. (2002), 'On properties of update sequences based on causal rejection', *Theory and Practice of Logic Programming* **2**(6), 711–767.

Eiter, T., Fink, M., Tompits, H. & Woltran, S. (2004), Simplifying logic programs under uniform and strong equivalence, *in* 'Logic Programming and Nonmonotonic Reasoning', Vol. 2923 of *Lecture Notes in Computer Science*, pp. 87–99.

Eiter, T., Ianni, G., Lukasiewicz, T. & Schindlauer, R. (2011), 'Well-founded semantics for description logic programs in the Semantic Web', *ACM Transactions on Computational Logic* **12**(2), 11:1–11:41.

Eiter, T., Ianni, G., Lukasiewicz, T., Schindlauer, R. & Tompits, H. (2008), 'Combining answer set programming with description logics for the Semantic Web', *Artificial Intelligence* **172**(12-13), 1495–1539.

Eiter, T., Ianni, G., Schindlauer, R. & Tompits, H. (2005), A uniform integration of higher-order reasoning and external evaluations in answer-set programming, *in* 'Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, IJCAI 2005', pp. 90–96.

Eiter, T., Ianni, G., Schindlauer, R., Tompits, H. & Wang, K. (2006), Forgetting in managing rules and ontologies, *in* 'Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence, WI 2006', pp. 411–419.

Eiter, T. & Wang, K. (2008), 'Semantic forgetting in answer set programming', *Artificial Intelligence* **172**(14), 1644–1672.

Fagin, R., Ullman, J. D. & Vardi, M. Y. (1983), On the semantics of updates in databases, *in* 'Proceedings of the Second ACM SIGACT-SIGMOD Symposium on Principles of Database Systems, PODS 1983', pp. 352–365.

Fermé, E., Krevneris, M. & Reis, M. (2008), 'An axiomatic characterization of ensconcement-based contraction', *Journal of Logic and Computation* **18**(5), 739–753.

Fitting, M. (1985), 'A Kripke-Kleene semantics for logic programs', *Journal of Logic Programming* **4**, 295–312.

Fuhrmann, A. (1991), 'Theory contraction through base contraction', *Journal of Philosophical Logic* **20**(2), 175–203.

Fuhrmann, A. & Hansson, S. O. (1994), 'A survey of multiple contractions', *Journal of Logic, Language and Information* **3**(1), 39–75.

Gärdenfors, P. (1981), 'An epistemic approach to conditionals', *American Philosophical Quarterly* **18**(3), 203–211.

Gärdenfors, P. (1988), *Knowledge in Flux: Modeling the Dynamics of Epistemic States*, MIT Press.

Gärdenfors, P. & Makinson, D. (1988), Revisions of knowledge systems using epistemic entrenchment, *in* 'Proceedings of the Second Conference on Theoretical Aspects of Reasoning About Knowledge, TARK 1988', pp. 83–95.

Gebser, M., Guziolowski, C., Ivanchev, M., Schaub, T., Siegel, A., Thiele, S. & Veber, P. (2010), Repair and prediction (under inconsistency) in large biological networks with answer set programming, *in* 'Principles of Knowledge Representation and Reasoning: Proceedings of the Twelfth International Conference, KR 2010', pp. 497–507.

Gelfond, M. & Lifschitz, V. (1988), The stable model semantics for logic programming, *in* 'Proceedings of the Fifth International Conference on Logic Programming', pp. 1070–1080.

Grasso, G., Leone, N. & Ricca, F. (2013), Answer set programming: Language, applications and development tools, *in* 'Web Reasoning and Rule Systems', Vol. 7994 of *Lecture Notes in Computer Science*, pp. 19–34.

Hansson, S. O. (1989), 'New operators for theory change', *Theoria* **55**(2), 114–132.

Hansson, S. O. (1991), 'Belief contraction without recovery', *Studia Logica* **50**(2), 251–260.

Hansson, S. O. (1993), 'Reversing the Levi identity', *Journal of Philosophical Logic* **22**(6), 637–669.

Hansson, S. O. (1995), 'Changes in preference', *Theory and Decision* **38**(1), 1–28.

Hansson, S. O. (1997), 'Semi-revision', *Journal of Applied Non-Classical Logics* **7**(1–2), 151–175.

Hansson, S. O. (1999), *A Textbook of Belief Dynamics. Theory Change and Database Updating*, Kluwer.

Hansson, S. O. & Wassermann, R. (2002), 'Local change', *Studia Logica* **70**(1), 49–76.

Harper, W. L. (1976), 'Rational conceptual change', *PSA: Proceedings of the Biennial Meeting of the Philosophy of Science Association* **Two: Symposia and Invited Papers**, 462–494.

Hitzler, P. & Parsia, B. (2009), Ontologies and rules, *in* 'Handbook on Ontologies', International Handbooks on Information Systems, pp. 111–132.

Inoue, K. & Sakama, C. (2004), Equivalence of logic programs under updates, *in* 'Logics in Artificial Intelligence', Vol. 3229 of *Lecture Notes in Computer Science*, pp. 174–186.

Katsuno, H. & Mendelzon, A. O. (1991), 'Propositional knowledge base revision and minimal change', *Artificial Intelligence* **52**(3), 263–294.

Katsuno, H. & Mendelzon, A. O. (1992), On the difference between updating a knowledge base and revising it, *in* P. Gärdenfors, ed., 'Belief revision', chapter 7, pp. 183–203.

Kharlamov, E., Zheleznyakov, D. & Calvanese, D. (2013), 'Capturing model-based ontology evolution at the instance level: The case of DL-Lite', *Journal of Computer and System Sciences* **79**(6), 835–872.

Knorr, M., Alferes, J. J. & Hitzler, P. (2011), 'Local closed world reasoning with description logics under the well-founded semantics', *Artificial Intelligence* **175**(9–10), 1528–1554.

Kowalski, R. (1974), Predicate logic as a programming language, *in* 'Proceedings of the IFIP Congress', pp. 569–574.

Krisnadhi, A., Maier, F. & Hitzler, P. (2011), OWL and rules, *in* 'Reasoning Web. Semantic Technologies for the Web of Data', Vol. 6848 of *Lecture Notes in Computer Science*, pp. 382–415.

Krümpelmann, P. & Kern-Isberner, G. (2012), Belief base change operations for answer set programming, *in* 'Logics in Artificial Intelligence', Vol. 7519 of *Lecture Notes in Computer Science*, pp. 294–306.

Lang, J., Liberatore, P. & Marquis, P. (2003), 'Propositional independence – Formula-variable independence and forgetting', *Journal of Artificial Intelligence Research* **18**, 391–443.

Leite, J. A. & Pereira, L. M. (1998), Generalizing updates: From models to programs, *in* 'Logic Programming and Knowledge Representation', Vol. 1471 of *Lecture Notes in Computer Science*, pp. 224–246.

Levi, I. (1977), Subjunctives, dispositions and chances, *in* 'Dispositions', Vol. 113 of *Synthese Library*, pp. 303–335.

Levi, I. (1980), *The Enterprise of Knowledge: An Essay on Knowledge, Credal Probability, and Chance*, MIT Press.

Lifschitz, V. (1991), Nonmonotonic databases and epistemic queries, *in* 'Proceedings of the Twelfth International Joint Conference on Artificial Intelligence, IJCAI 1991', Vol. 1, pp. 381–386.

Lifschitz, V., Pearce, D. & Valverde, A. (2001), 'Strongly equivalent logic programs', *ACM Transactions on Computational Logic* **2**(4), 526–541.

Lifschitz, V., Pearce, D. & Valverde, A. (2007), A characterization of strong equivalence for logic programs with variables, *in* 'Logic Programming and Nonmonotonic Reasoning', Vol. 4483 of *Lecture Notes in Computer Science*, pp. 188–200.

Lin, F. & Reiter, R. (1994), Forget it!, *in* 'Proceedings of the AAAI Fall Symposium on Relevance', pp. 154–159.

Liu, H., Lutz, C., Miličić, M. & Wolter, F. (2011), 'Foundations of instance level updates in expressive description logics', *Artificial Intelligence* **175**(18), 2170–2197.

Lloyd, J. W. (1987), *Foundations of Logic Programming*, Springer-Verlag New York.

Łukasiewicz, J. (1941), 'Die Logik und das Grundlagenproblem', *Les Entries de Zürich sur les Fondaments et la Méthode des Sciences Mathématiques* pp. 82–108.

Lukasiewicz, T. (2010), 'A novel combination of answer set programming with description logics for the Semantic Web', *IEEE Transactions on Knowledge and Data Engineering* **22**(11), 1577–1592.

Makinson, D. (1987), 'On the status of the postulate of recovery in the logic of theory change', *Journal of Philosophical Logic* **16**(4), 383–394.

Makinson, D. (1997), 'Screened revision', *Theoria* **63**(1–2), 14–23.

McCarthy, J. (1958), Programs with common sense, *in* 'Proceedings of the Symposium on Mechanisation of Thought Processes', pp. 77–84.

Motik, B. & Rosati, R. (2010), 'Reconciling description logics and rules', *Journal of the ACM* **57**(5), 30:1–30:62.

Nayak, A. C. (1994), 'Foundational belief change', *Journal of Philosophical Logic* **23**(5), 495–533.

Nayak, A. C., Nelson, P. & Polansky, H. (1996), 'Belief change as change in epistemic entrenchment', *Synthese* **109**(2), 143–174.

Niederée, R. (1991), Multiple contraction. A further case against Gärdenfors' principle of recovery, *in* 'The Logic of Theory Change', Vol. 465 of *Lecture Notes in Computer Science*, pp. 322–334.

Osorio, M. & Cuevas, V. (2007), 'Updates in answer set programming: An approach based on basic structural properties', *Theory and Practice of Logic Programming* **7**(4), 451–479.

Parikh, R. (1999), 'Beliefs, belief revision, and splitting languages', *Logic, Language and Computation* **2**, 266–278.

Pearce, D., Tompits, H. & Woltran, S. (2009), 'Characterising equilibrium logic and nested logic programs: Reductions and complexity', *Theory and Practice of Logic Programming* **9**(5), 565–616.

Pearce, D. & Valverde, A. (2008), Quantified equilibrium logic and foundations for answer set programs, *in* 'Logic Programming', Vol. 5366 of *Lecture Notes in Computer Science*, pp. 546–560.

Qi, G. & Du, J. (2009), Model-based revision operators for terminologies in description logics, *in* 'Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence, IJCAI 2009', pp. 891–897.

Rosati, R. (2006), $\mathcal{DL}$+log: Tight integration of description logics and disjunctive Datalog, *in* 'Principles of Knowledge Representation and Reasoning: Proceedings of the Tenth International Conference, KR 2006', pp. 68–78.

Rott, H. (1992), Modellings for belief change: Base contraction, multiple contraction, and epistemic entrenchment (Preliminary report), *in* 'Logics in Artificial Intelligence', Vol. 633 of *Lecture Notes in Computer Science*, pp. 139–153.

Rudolph, S. (2011), Foundations of description logics, *in* 'Reasoning Web. Semantic Technologies for the Web of Data', Vol. 6848 of *Lecture Notes in Computer Science*, pp. 76–136.

Satoh, K. (1988), Nonmonotonic reasoning by minimal belief revision, *in* 'Proceedings of the International Conference on Fifth Generation Computer Systems', pp. 455–462.

Schaub, T. & Wang, K. (2003), 'A semantic framework for preference handling in answer set programming', *Theory and Practice of Logic Programming* **3**(4–5), 569–607.

Schwind, N. & Inoue, K. (2013), Characterization theorems for revision of logic programs, *in* 'Logic Programming and Nonmonotonic Reasoning', Vol. 8148 of *Lecture Notes in Computer Science*, pp. 485–498.

Shen, Y.-D. & Wang, K. (2011), Extending logic programs with description logic expressions for the Semantic Web, *in* 'The Semantic Web – ISWC 2011', Vol. 7031 of *Lecture Notes in Computer Science*, pp. 633–648.

Slota, M. (2012), Updates of hybrid knowledge bases, PhD thesis, Universidade Nova de Lisboa.

Slota, M. & Leite, J. (2012), Robust equivalence models for semantic updates of answer-set programs, *in* 'Principles of Knowledge Representation and Reasoning: Proceedings of the Thirteenth International Conference, KR 2012', pp. 158–168.

Slota, M. & Leite, J. (2014), 'The rise and fall of semantic rule updates based on SE-models', *Theory and Practice of Logic Programming* **14**(6), 869–907.

Slota, M., Leite, J. & Swift, T. (2011), 'Splitting and updating hybrid knowledge bases', *Theory and Practice of Logic Programming* **11**(4–5), 801–819.

Turner, H. (2003), 'Strong equivalence made easy: Nested expressions and weight constraints', *Theory and Practice of Logic Programming* **3**(4), 609–622.

Wang, Y., You, J.-H., Yuan, L. Y., Shen, Y.-D. & Zhang, M. (2012), 'The loop formula based semantics of description logic programs', *Theoretical Computer Science* **415**, 60–85.

Wang, Y., Zhang, Y., Zhou, Y. & Zhang, M. (2014), 'Knowledge forgetting in answer set programming', *Journal of Artificial Intelligence Research* **50**(1), 31–70.

Wang, Z., Wang, K. & Topor, R. (2015), 'DL-Lite ontology revision based on an alternative semantic characterization', *ACM Transactions on Computational Logic* **16**(4), 31:1–31:37.

Wang, Z., Wang, K., Topor, R. & Pan, J. (2008), Forgetting concepts in DL-Lite, *in* 'The Semantic Web: Research and Applications', Vol. 5021 of *Lecture Notes in Computer Science*, pp. 245–257.

Wang, Z., Wang, K., Topor, R. & Pan, J. (2010), 'Forgetting for knowledge bases in DL-Lite', *Annals of Mathematics and Artificial Intelligence* **58**(1), 117–151.

Wang, Z., Wang, K. & Topor, R. W. (2010*a*), A new approach to knowledge base revision in DL-Lite, *in* 'Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010', pp. 369–374.

Wang, Z., Wang, K. & Topor, R. W. (2010*b*), Revising general knowledge bases in description logics, *in* 'Principles of Knowledge Representation and Reasoning: Proceedings of the Twelfth International Conference, KR 2010', pp. 599–601.

Wassermann, R. (2000), Resource-bounded belief revision, PhD thesis, Universiteit van Amsterdam.

Wassermann, R. (2011), 'On AGM for non-classical logics', *Journal of Philosophical Logic* **40**(2), 271–294.

Williams, M.-A. (1994), On the logic of theory base change, *in* 'Logics in Artificial Intelligence', Vol. 838 of *Lecture Notes in Computer Science*, pp. 86–105.

Winslett, M. (1990), *Updating Logical Databases (Cambridge Tracts in Theoretical Computer Science)*, Cambridge University Press.

Wong, K.-S. (2008), 'Sound and complete inference rules for SE-consequence', *Journal of Artificial Intelligence Research* **31**, 205–216.

Zhang, Y. & Foo, N. Y. (2006), 'Solving logic program conflict through strong and weak forgettings', *Artificial Intelligence* **170**(8–9), 739–778.

Zheleznyakov, D., Calvanese, D., Kharlamov, E. & Nutt, W. (2010), Updating TBoxes in DL-Lite, *in* 'Proceedings of the 23rd International Workshop on Description Logics (DL 2010)', Vol. 573 of *CEUR Workshop Proceedings*.

Zhuang, Z., Wang, Z., Wang, K. & Delgrande, J. (2015), Extending AGM contraction to arbitrary logics, *in* 'Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015', pp. 3299–3307.

Zhuang, Z., Wang, Z., Wang, K. & Qi, G. (2014), Contraction and revision over DL-Lite TBoxes, *in* 'Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2014', pp. 1149–1155.

A

Proofs for Chapter 4

Proposition 4.1: Let $\dot{-}$ be a contraction operator on $\mathcal{LP}_{\mathcal{A}}$. If $\dot{-}$ satisfies ($\dot{-}$3b), then it satisfies ($\dot{-}$8b).

Proof. Proof by contrapositive: Let $r \in P \setminus (P \dot{-} Q)$. Assume $SE(P \dot{-} Q) \subseteq SE(Q) \cup SE(r)$. Then for all P' such that $P \dot{-} Q \subseteq P' \subseteq P$ with $SE(P') \nsubseteq SE(Q)$: $SE(P') \cap SE(r) \nsubseteq SE(Q)$, due to the assumption and due to $SE(P') \subseteq SE(P \dot{-} Q)$. □

B

Proofs for Chapter 5

Theorem 5.5: The revision operator $*_\gamma$ satisfies $(*1)$–$(*6)$.

Proof. $(*1)$: Follows directly from Definition 5.3.
$(*2)$: Follows directly from Definition 5.3.
$(*3)$: If Q is not satisfiable, then $P*_\gamma Q = P+Q$. Otherwise, since $\bigcap \gamma(\mathbb{P}_Q) \subseteq P$ we have $\bigcap \gamma(\mathbb{P}_Q) + Q \subseteq P + Q$.
$(*4)$: If $P + Q$ is satisfiable, then $\mathbb{P}_Q = \{P\} = \gamma(\mathbb{P}_Q)$, for any selection function γ, and thus $P *_\gamma Q = P + Q$.
$(*5)$: If Q is not satisfiable, then $P *_\gamma Q = P + Q$ is not satisfiable. If Q is satisfiable, then for any $R \in \mathbb{P}_Q$, $R + Q$ is satisfiable, which implies $P *_\gamma Q$ is satisfiable.
$(*6)$: Follows directly from Definition 5.3. □

Lemma 5.9: Let $P, Q \in \mathcal{LP}_\mathcal{A}$ and γ be a selection function for P. If $r \in P \cap Q$, then $r \in \bigcap \gamma(\mathbb{P}_Q)$.

Proof. Let $P, Q \in \mathcal{LP}_\mathcal{A}$. Assume there exists $r \in (P \cap Q) \setminus (\bigcap \gamma(\mathbb{P}_Q))$. Then there exists $R \in \gamma(\mathbb{P}_Q) : r \notin R$. It follows that $SE(R \cup \{r\}) \cap SE(Q) \neq \emptyset$ since $SE(R) \cap SE(Q) \neq \emptyset$ by Definition 5.1 and $r \in Q$ implies $SE(Q) \subseteq SE(r)$. This is a contradiction because R is maximal by Definition 5.1. □

Theorem 5.10: Let $P \in \mathcal{LP}_{\mathcal{A}}$. An operator $*_\gamma$ is a partial meet revision operator for P generated by some selection function γ for P iff $*_\gamma$ satisfies ($*$1b)–($*$5b).

Proof. We first show that a partial meet revision operator $*_\gamma$ for P generated by some selection function γ for P satisfies ($*$1b)–($*$5b).
($*$1b): Since ($*$1b) = ($*$2) and $*_\gamma$ satisfies ($*$2), $*_\gamma$ also satisfies ($*$1b).
($*$2b): Since ($*$2b) = ($*$3) and $*_\gamma$ satisfies ($*$3), $*_\gamma$ also satisfies ($*$2b).
($*$3b): Let $r \in P$. Assume that for all P' with $P *_\gamma Q \subseteq P' \subset P + Q$ and P' being satisfiable, it holds that $P' \cup \{r\}$ is satisfiable. In particular, for each $R \in \mathbb{P}_Q$ with $P *_\gamma Q \subseteq R \cup Q$, this implies $R \cup Q \cup \{r\}$ is satisfiable. As each R is subset-maximal, it follows that $r \in R$ and thus $r \in \bigcap \gamma(\mathbb{P}_Q)$. From Definition 5.3 we can then conclude $r \notin P \setminus (P *_\gamma Q)$.
($*$4b): For all $P' \subseteq P$, let $P' + Q$ be satisfiable iff $P' + R$ is satisfiable. Then $\mathbb{P}_Q = \mathbb{P}_R$ by Definition 5.1 and so $\bigcap \gamma(\mathbb{P}_Q) = \bigcap \gamma(\mathbb{P}_R)$ as well as $P \cap \bigcap \gamma(\mathbb{P}_Q) = P \cap \bigcap \gamma(\mathbb{P}_R)$. By Lemma 5.9 we obtain $(P \cap \bigcap \gamma(\mathbb{P}_Q)) \cup (P \cap Q) = (P \cap \bigcap \gamma(\mathbb{P}_R)) \cup (P \cap R)$. This means $P \cap (\bigcap \gamma(\mathbb{P}_Q) \cup Q) = P \cap (\bigcap \gamma(\mathbb{P}_R) \cup R)$. Thus, $P \cap (P *_\gamma Q) = P \cap (P *_\gamma R)$.
($*$5b): If Q is satisfiable, then for any $R \in \mathbb{P}_Q$, $R + Q$ is satisfiable, which implies $P *_\gamma Q$ is satisfiable.

We now show that any operator $\circ_\gamma$ for P satisfying ($*$1b)–($*$5b) is a partial meet revision operator for P generated by some selection function for P. We first find a selection function γ for P. Let γ be such that (i) if $\mathbb{P}_Q = \emptyset$, then $\gamma(\mathbb{P}_Q) = \emptyset$ and (ii) $\gamma(\mathbb{P}_Q) = \{ R \in \mathbb{P}_Q \mid P \cap (P \circ_\gamma Q) \subseteq R \}$ otherwise.

We begin by showing that γ is a function. If $\mathbb{P}_Q = \mathbb{P}_R$, then $P \cap (P \circ_\gamma Q) = P \cap (P \circ_\gamma R)$ by ($*$4b). This means $\gamma(\mathbb{P}_Q) = \gamma(\mathbb{P}_R)$ according to our definition of γ.

We next show that γ is a selection function. Clearly, $\gamma(\mathbb{P}_Q) \subseteq \mathbb{P}_Q$ by our definition of γ. If $\mathbb{P}_Q \neq \emptyset$, then Q is satisfiable by Definition 5.1 and thus $P \circ_\gamma Q$ is satisfiable by ($*$5b). Since $Q \subseteq P \circ_\gamma Q$ by ($*$1b) and $P \circ_\gamma Q \subseteq P \cup Q$ by ($*$2b), it follows that $(P \cap (P \circ_\gamma Q)) \cup Q$ is satisfiable. This means that there exists $R \in \mathbb{P}_Q$ such that $P \cap (P \circ_\gamma Q) \subseteq R$. From our definition of γ we therefore obtain that $\gamma(\mathbb{P}_Q) \neq \emptyset$.

Finally, we show that $\circ_\gamma$ is a partial meet revision operator for P, that is, $P \circ_\gamma Q = P \cup Q$ if Q is not satisfiable and $P \circ_\gamma Q = \bigcap \gamma(\mathbb{P}_Q) \cup Q$ otherwise. Consider first the limiting case that Q is not satisfiable. If $r \in P \setminus (P \circ_\gamma Q)$,

then there exists P' such that $P \circ_\gamma Q \subseteq P' \subset P \cup Q$ and P' is satisfiable but $P' \cup \{r\}$ is not satisfiable by ($*$3b). This is a contradiction since $Q \subseteq P'$ by ($*$1b). Therefore, it holds for all $r \in P$ that $r \in P \circ_\gamma Q$, that is, $P \subseteq P \circ_\gamma Q$. Since $Q \subseteq P \circ_\gamma Q$ by ($*$1b) and $P \circ_\gamma Q \subseteq P \cup Q$ by ($*$2b), we can conclude $P \circ_\gamma Q = P \cup Q$.

Assume now that Q is satisfiable. Let $r \in P \setminus (P \circ_\gamma Q)$. If $\mathbb{P}_Q = \emptyset$, then it follows from ($*$1b) and ($*$3b) that $P \circ_\gamma Q = Q$. Since $\gamma(\mathbb{P}_Q) = \emptyset$ by our definition of γ, we thus have $P \circ_\gamma Q = Q = \bigcap \gamma(\mathbb{P}_Q) \cup Q$. If $\mathbb{P}_Q \neq \emptyset$, then it follows directly from our definition of γ that $P \cap (P \circ_\gamma Q) \subseteq \bigcap \gamma(\mathbb{P}_Q)$. From ($*$1b) and ($*$2b) we then obtain $P \circ_\gamma Q \subseteq \bigcap \gamma(\mathbb{P}_Q) \cup Q$. To show the converse inclusion, first assume the case that $P \cup Q$ is satisfiable. This implies that for any $P' \subseteq P \cup Q$ it holds that P' is satisfiable. Applying ($*$3b), we obtain $P \setminus (P \circ_\gamma Q) = \emptyset$ and thus $P \subseteq P \circ_\gamma Q$. From ($*$1b) and ($*$2b) it follows that $P \circ_\gamma Q = P \cup Q$. Moreover, due to the assumption that $P \cup Q$ is satisfiable and Definition 5.1, we have $\mathbb{P}_Q = \{P\}$. By our definition of γ, we obtain $\gamma(\mathbb{P}_Q) = \{P\}$ and thus $\bigcap \gamma(\mathbb{P}_Q) = P$ and can conclude $P \circ_\gamma Q = \bigcap \gamma(\mathbb{P}_Q) \cup Q$. Lastly, assume the case that $P \cup Q$ is not satisfiable. We will show that $r \notin P \circ_\gamma Q$ implies $r \notin \bigcap \gamma(\mathbb{P}_Q) \cup Q$. If $r \notin P$, then $r \notin (P \circ_\gamma Q) \setminus Q$ by ($*$1b) and ($*$2b) and $r \notin \bigcap \gamma(\mathbb{P}_Q)$ by Definition 5.1. Since $r \notin P \circ_\gamma Q$ implies $r \notin Q$ by ($*$1b), it follows that $r \notin ((P \circ_\gamma Q) \setminus Q) \cup Q) = P \circ_\gamma Q$ and $r \in \bigcap \gamma(\mathbb{P}_Q) \cup Q$. Now assume $r \in P \setminus (P \circ_\gamma Q)$. According to ($*$3b), then there exists P' such that $P \circ_\gamma Q \subseteq P' \subset P \cup Q$ and P' is satisfiable but $P' \cup \{r\}$ is not satisfiable. This means that there exists $R \in \mathbb{P}_Q$ such that $P \cap P' \subseteq R$ and $r \notin R$. Since $P \cap (P \circ_\gamma Q) \subseteq P \cap P' \subseteq R$, we obtain from our definition of γ that $R \in \gamma(\mathbb{P}_Q)$. We can thus conclude from $r \notin R$ that $r \notin \bigcap \gamma(\mathbb{P}_Q)$. □

Theorem 5.13: The contraction operator $\dot{-}_\gamma$ satisfies ($\dot{-}$1)–($\dot{-}$4) and ($\dot{-}$6).

Proof. ($\dot{-}$1): Follows directly from Definition 5.11.
($\dot{-}$2): Follows directly from Definition 5.11.
($\dot{-}$3): If $P \not\models_s Q$, then $\mathbb{P}_Q^- = \{P\} = \gamma(\mathbb{P}_Q^-)$, for any selection function γ, and thus $P \dot{-}_\gamma Q = P$.
($\dot{-}$4): Let $\not\models_s Q$. For any $R \in \mathbb{P}_Q^-$, $R \not\models_s Q$, which implies $P \dot{-}_\gamma Q \not\models_s Q$.
($\dot{-}$6): Follows directly from Definition 5.11. □

Lemma 5.14: Let $P \in \mathcal{LP}_\mathcal{A}$. For any $Q, R \in \mathcal{LP}_\mathcal{A}$, it holds that $\mathbb{P}_{Q+R}^- \subseteq \mathbb{P}_Q^- \cup \mathbb{P}_R^-$.

Proof. Let $P, Q, R \in \mathcal{LP}_{\mathcal{A}}$. It follows from $\overline{SE(Q+R)} = \overline{SE(Q)} \cup \overline{SE(R)}$ and the definition of $\mathbb{P}^-_{Q+R}$ that $\mathbb{P}^-_{Q+R} = \{ S \in \mathbb{P}^-_Q \cup \mathbb{P}^-_R \mid S \not\subset S' \text{ for any } S' \in \mathbb{P}^-_Q \cup \mathbb{P}^-_R \}$. Thus, $\mathbb{P}^-_{Q+R} \subseteq \mathbb{P}^-_Q \cup \mathbb{P}^-_R$. □

Lemma 5.15: Let $P \in \mathcal{LP}_{\mathcal{A}}$ and γ' be determined by a maximised transitive relation. For any $Q, R \in \mathcal{LP}_{\mathcal{A}}$, it holds that $\gamma'(\mathbb{P}^-_{Q+R}) \subseteq \gamma'(\mathbb{P}^-_Q) \cup \gamma'(\mathbb{P}^-_R)$.

Proof. Let $P, Q, R \in \mathcal{LP}_{\mathcal{A}}$. Assume there exists $S \in \gamma'(\mathbb{P}^-_{Q+R}) : S \notin \gamma'(\mathbb{P}^-_Q) \cup \gamma'(\mathbb{P}^-_R)$. Then $S \notin \gamma'(\mathbb{P}^-_Q)$ and $S \notin \gamma'(\mathbb{P}^-_R)$. Case 1: If $S \notin \mathbb{P}^-_Q$, this means that $S \in \mathbb{P}^-_R$ by Lemma 5.14. From Definition 5.6 it follows that there exists $S' \in \mathbb{P}^-_R : S \subset S'$, a contradiction since $S \in \mathbb{P}^-_{Q+R}$. Case 2: $S \in \mathbb{P}^-_Q$. Follows analogously as Case 1. □

Theorem 5.16: Let γ' be determined by a maximised transitive relation. The contraction operator $\dot{-}_{\gamma'}$ satisfies ($\dot{-}$7).

Proof. Let $P, Q, R \in \mathcal{LP}_{\mathcal{A}}$ and $r \in (P \dot{-}_{\gamma'} Q) \cap (P \dot{-}_{\gamma'} R)$. This means that $r \in \bigcap \gamma'(\mathbb{P}^-_Q)$ and $r \in \bigcap \gamma'(\mathbb{P}^-_R)$. By Lemma 5.15, we have for all $S \in \gamma'(\mathbb{P}^-_{Q+R}) : r \in S$, so that $r \in \bigcap \gamma'(\mathbb{P}^-_{Q+R})$. Thus, $(P \dot{-}_{\gamma'} Q) \cap (P \dot{-}_{\gamma'} R) \subseteq P \dot{-}_{\gamma'} (Q+R)$. □

Theorem 5.18: Let $P \in \mathcal{LP}_{\mathcal{A}}$. An operator $\dot{-}_{\gamma}$ is a partial meet contraction operator for P generated by some selection function γ for P iff $\dot{-}_{\gamma}$ satisfies ($\dot{-}$1b)–($\dot{-}$4b).

Proof. We first show that a partial meet contraction operator $\dot{-}_{\gamma}$ for P generated by some selection function γ for P satisfies ($\dot{-}$1b)–($\dot{-}$4b).
($\dot{-}$1b): Follows from ($\dot{-}$1b) = ($\dot{-}$2) and satisfaction of ($\dot{-}$2).
($\dot{-}$2b): Follows from ($\dot{-}$2b) = ($\dot{-}$4) and satisfaction of ($\dot{-}$4).
($\dot{-}$3b): Let $r \in P$. Assume that for all P' with $P \dot{-}_{\gamma} Q \subseteq P' \subset P$ and $P' \not\models_s Q$, it holds that $P' \cup \{r\} \not\models_s Q$. In particular, for each $R \in \mathbb{P}^-_Q$ with $P \dot{-}_{\gamma} Q \subseteq R$, this implies $R \cup \{r\} \not\models_s Q$. As each R is subset-maximal by definition of $\mathbb{P}^-_Q$, it follows that $r \in R$ and thus $r \in P \dot{-}_{\gamma} Q$.
($\dot{-}$4b): For all $P' \subseteq P$, let $P' \not\models_s Q$ iff $P' \not\models_s R$. Then $\mathbb{P}^-_Q = \mathbb{P}^-_R$ by definition of $\mathbb{P}^-_Q$ and so $\gamma(\mathbb{P}^-_Q) = \gamma(\mathbb{P}^-_R)$ as well as $\bigcap \gamma(\mathbb{P}^-_Q) = \bigcap \gamma(\mathbb{P}^-_R)$. Thus, $P \dot{-}_{\gamma} Q = P \dot{-}_{\gamma} R$ by Definition 5.11.

We now show that any operator $\circ_{\gamma}$ for P satisfying ($\dot{-}$1b)–($\dot{-}$4b) is a partial meet contraction operator for P generated by some selection function for P. We first find a selection function γ for P. Let γ be such that (i) if $\mathbb{P}^-_Q = \emptyset$, then $\gamma(\mathbb{P}^-_Q) = \emptyset$ and (ii) $\gamma(\mathbb{P}^-_Q) = \{ R \in \mathbb{P}^-_Q \mid P \circ_{\gamma} Q \subseteq R \}$ otherwise.

We begin by showing that γ is a function. If $\mathbb{P}_Q^- = \mathbb{P}_R^-$, then $P \circ_\gamma Q = P \circ_\gamma R$ by ($\dot{-}$4b). This means $\gamma(\mathbb{P}_Q^-) = \gamma(\mathbb{P}_R^-)$ according to our definition of γ.

We next show that γ is a selection function. Clearly, $\gamma(\mathbb{P}_Q^-) \subseteq \mathbb{P}_Q^-$ by our definition of γ. If $\mathbb{P}_Q^- \neq \emptyset$, then $\not\models_s Q$ by the definition of $\mathbb{P}_Q^-$ and thus $P \circ_\gamma Q \not\models_s Q$ by ($\dot{-}$2b). It follows from $P \circ_\gamma Q \subseteq P$ due to ($\dot{-}$1b) that there exists $R \in \mathbb{P}_Q^-$ such that $P \circ_\gamma Q \subseteq R$. From our definition of γ we therefore obtain that $\gamma(\mathbb{P}_Q^-) \neq \emptyset$.

Finally, we show that $\circ_\gamma$ is a partial meet contraction operator for P, that is, $P \circ_\gamma Q = P$ if $\models_s Q$ and $P \circ_\gamma Q = \bigcap \gamma(\mathbb{P}_Q^-)$ otherwise. Consider first the limiting case that $\models_s Q$. If $r \in P \setminus (P \circ_\gamma Q)$, then there exists P' such that $P \circ_\gamma Q \subseteq P' \subset P$ and $P' \not\models_s Q$ but $P' \cup \{r\} \models_s Q$ by ($\dot{-}$3b). This is a contradiction since $\models_s Q$. Therefore, it holds for all $r \in P$ that $r \in P \circ_\gamma Q$, that is, $P \subseteq P \circ_\gamma Q$. Since $P \circ_\gamma Q \subseteq P$ by ($\dot{-}$1b), we can conclude $P \circ_\gamma Q = P$.

Assume now that $\not\models_s Q$. Let $r \in P \setminus (P \circ_\gamma Q)$. If $\mathbb{P}_Q^- = \emptyset$, then it follows from ($\dot{-}$2b) and ($\dot{-}$3b) that $P \circ_\gamma Q = \emptyset$. Since $\gamma(\mathbb{P}_Q^-) = \emptyset$ by our definition of γ, we thus have $P \circ_\gamma Q = \bigcap \gamma(\mathbb{P}_Q^-)$. If $\mathbb{P}_Q^- \neq \emptyset$, then it follows directly from our definition of γ that $P \circ_\gamma Q \subseteq \bigcap \gamma(\mathbb{P}_Q^-)$. To show the converse inclusion, first assume the case that $P \not\models_s Q$. This implies that for any $P' \subseteq P$ it holds that $P' \not\models_s Q$. Applying ($\dot{-}$3b), we obtain $P \setminus (P \circ_\gamma Q) = \emptyset$ and thus $P \subseteq P \circ_\gamma Q$. From ($\dot{-}$1b) it follows that $P \circ_\gamma Q = P$. Moreover, due to the assumption that $P \not\models_s Q$ and the definition of $\mathbb{P}_Q^-$, we have $\mathbb{P}_Q^- = \{P\}$. By our definition of γ, we obtain $\gamma(\mathbb{P}_Q^-) = \{P\}$ and thus $\bigcap \gamma(\mathbb{P}_Q^-) = P$ and can conclude $P \circ_\gamma Q = \bigcap \gamma(\mathbb{P}_Q^-)$. Lastly, assume the case that $P \models_s Q$. We will show that $r \notin P \circ_\gamma Q$ implies $r \notin \bigcap \gamma(\mathbb{P}_Q^-)$. If $r \notin P$, then $r \notin P \circ_\gamma Q$ by ($\dot{-}$1b) and $r \notin \bigcap \gamma(\mathbb{P}_Q^-)$ by the definition of $\mathbb{P}_Q^-$. Now assume $r \in P \setminus (P \circ_\gamma Q)$. According to ($\dot{-}$3b), then there exists P' such that $P \circ_\gamma Q \subseteq P' \subset P$ and $P' \not\models_s Q$ but $P' \cup \{r\} \models_s Q$. This means that there exists $R \in \mathbb{P}_Q^-$ such that $P' \subseteq R$ and $r \notin R$. Since $P \circ_\gamma Q \subseteq P' \subseteq R$, we obtain from our definition of γ that $R \in \gamma(\mathbb{P}_Q^-)$. We can thus conclude from $r \notin R$ that $r \notin \bigcap \gamma(\mathbb{P}_Q^-)$. □

Proposition 5.19: The contraction operator $\dot{-}_\gamma$ satisfies ($\dot{-}$5b), ($\dot{-}$6b), and ($\dot{-}$8b).

Proof. ($\dot{-}$5b): Since ($\dot{-}$5b) = ($\dot{-}$3) and $\dot{-}_\gamma$ satisfies ($\dot{-}$3), $\dot{-}_\gamma$ also satisfies ($\dot{-}$5b).
($\dot{-}$6b): Since ($\dot{-}$6b) = ($\dot{-}$6) and $\dot{-}_\gamma$ satisfies ($\dot{-}$6), $\dot{-}_\gamma$ also satisfies ($\dot{-}$6b).
($\dot{-}$8b): Follows from satisfaction of ($\dot{-}$3b) and Proposition 4.1. □

Proposition 5.21: Let $P \in \mathcal{LP}_{\mathcal{A}}$, γ be a selection function for P, and $*$ an operator for P such that for any $Q \in \mathcal{LP}_{\mathcal{A}}$: $P * Q = (P \dot{-}_{\gamma} \overline{Q}) + Q$. Then $P * Q = P *_{\gamma} Q$.

Proof. Let $\overline{SE(Q)} = \mathcal{SE}$. Then $P \dot{-}_{\gamma} \overline{Q} = P$ by Definition 5.11 and thus $P * Q = (P \dot{-}_{\gamma} \overline{Q}) + Q = P + Q = P *_{\gamma} Q$ by Definition 5.3. Otherwise, $\overline{SE(Q)} \neq \mathcal{SE}$ such that $\mathbb{P}^{-}_{\overline{Q}} = \{ R \subseteq P \mid SE(R) \cap SE(Q) \neq \emptyset$ and for all R' with $R \subset R' \subseteq P : SE(R') \cap SE(Q) = \emptyset \} = \mathbb{P}_Q$ by Definition 5.1. It follows that $P * Q = (P \dot{-}_{\gamma} \overline{Q}) + Q = \bigcap \gamma(\mathbb{P}_Q) + Q = P *_{\gamma} Q$ by Definitions 5.3 and 5.11. □

Proposition 5.22: Let $P \in \mathcal{LP}_{\mathcal{A}}$, γ be a selection function for P, and $\dot{-}$ an operator for P such that for any $Q \in \mathcal{LP}_{\mathcal{A}}$: $P \dot{-} Q = P \cap (P *_{\gamma} \overline{Q})$. Then $P \dot{-} Q = P \dot{-}_{\gamma} Q$.

Proof. Let $\overline{SE(Q)} = \emptyset$. Then $P *_{\gamma} \overline{Q} = P + \overline{Q}$ by Definition 5.3 and thus $P \dot{-} Q = P \cap (P *_{\gamma} \overline{Q}) = P = P \dot{-}_{\gamma} Q$ by Definition 5.11. Otherwise, $\overline{SE(Q)} \neq \emptyset$ such that $\mathbb{P}_{\overline{Q}} = \{ R \subseteq P \mid SE(R) \cap \overline{SE(Q)} \neq \emptyset$ and for all R' with $R \subset R' \subseteq P : SE(R') \cap \overline{SE(Q)} = \emptyset \} = \mathbb{P}^{-}_{Q}$ by definition of $\mathbb{P}^{-}_{Q}$. It follows that $P \dot{-} Q = P \cap (P *_{\gamma} \overline{Q}) = P \cap (\bigcap \gamma(\mathbb{P}^{-}_{Q}) + \overline{Q})$ by Definition 5.3. Assume there exists a rule $r \in \overline{Q}$ with $r \in P \setminus \bigcap \gamma(\mathbb{P}^{-}_{Q})$. Then there exists an $R \in \gamma(\mathbb{P}^{-}_{Q}) : r \notin R$, a contradiction since $SE(\overline{Q}) = \overline{SE(Q)} \subseteq SE(r)$ and R is maximal. We therefore obtain $P \dot{-} Q = P \dot{-}_{\gamma} Q$ by Definition 5.11. □

Proposition 5.26: Let $P, Q \in \mathcal{LP}_{\mathcal{A}}$ and $SE(P) \neq \emptyset \neq SE(Q)$. For any $R \subseteq P$, if $SE(R) \cap SE(Q) = \emptyset$ and for all $R' \subset R : SE(R') \cap SE(Q) \neq \emptyset$, then there exists $\mathbb{M} \in 2^{\mathcal{M}(P)|_Q}$ such that $R \subseteq \bigcup \mathbb{M}$.

Proof. Let P, Q be satisfiable logic programs and $R \subseteq P$ such that $SE(R) \cap SE(Q) = \emptyset$ and for each $R' \subset R : SE(R') \cap SE(Q) \neq \emptyset$. Then there exists some $a_j \in \mathcal{A}$ such that $a_j \in At(Q)$ and there exist one or more rules $r_i \in R$ for each a_j such that $a_j \in At(r_i)$. For each r_i, there exists a corresponding r_i-module $M(P)^{r_i}|_{a_j}$ including a_j, such that $r_i \in M(P)^{r_i}|_{a_j}$. It follows from Definition 5.23 that for all remaining rules $r' \in R \setminus r_i : r' \in \bigcup_{i,j} M(P)^{r_i}|_{a_j}$. □

Corollary 5.27: Let $P, Q \in \mathcal{LP}_{\mathcal{A}}$ and $SE(P) \neq \emptyset$. Then $SE(P) \cap SE(Q) = \emptyset$ if and only if $SE\left(\bigcup \mathcal{M}(P)|_Q\right) \cap SE(Q) = \emptyset$.

Proof. "If": Since $SE(P) \subseteq SE(\bigcup \mathcal{M}(P)|_Q)$, if $SE(\bigcup \mathcal{M}(P)|_Q) \cap SE(Q) = \emptyset$, then also $SE(P) \cap SE(Q) = \emptyset$.

"Only if": Follows from Proposition 5.26 if Q is satisfiable. Trivial if Q is not satisfiable. □

Theorem 5.28: For any $P, Q \in \mathcal{LP}_{\mathcal{A}}$, let $P \setminus \mathcal{M}(P)|_Q = \{ r \in P \mid \forall M \in \mathcal{M}(P)|_Q : r \notin M \}$ and $\mathcal{M}(P)|_Q^{\circ}$ denote the output of Algorithm 1 for the inputs $\mathcal{M}(P)|_Q$, $\circ \in \{*_\gamma, \dot{-}_\gamma\}$, and Q. Then $P *_\gamma Q = P \setminus \mathcal{M}(P)|_Q + \bigcup \mathcal{M}(P)|_Q^{*_\gamma} + Q$ (or $P \dot{-}_\gamma Q = P \setminus \mathcal{M}(P)|_Q + \bigcup \mathcal{M}(P)|_Q^{\dot{-}_\gamma}$, respectively) for some selection function γ for P.

Proof. To prove the equation for revision, we need to show that $P \setminus \mathcal{M}(P)|_Q + \bigcup \mathcal{M}(P)|_Q^{*_\gamma} = \bigcap \gamma(\mathbb{P}_Q)$ for some γ. Let $Z \subseteq P$ be the set of rules that are eliminated during the revision operation, i.e., $P *_\gamma Q = \bigcap \gamma(\mathbb{P}_Q) + Q = P \setminus Z + Q$, and let $Z' \subseteq P$ be the set of rules that are eliminated by ModChange.

We first show that $Z \subseteq Z'$. Assume that $Z' = \emptyset$ until the last iteration of the while-loop. In the last iteration, we have $n = |\mathcal{M}(P)|_Q|$ and ModChange computes $\bigcup \mathcal{M}(P)|_Q *_\gamma Q = \bigcup \mathcal{M}(P)|_Q^{*_\gamma}$. Since $At(P \setminus \mathcal{M}(P)|_Q) \cap At(\bigcup \mathcal{M}(P)|_Q) = \emptyset$, it holds that $P \setminus \mathcal{M}(P)|_Q + \bigcup \mathcal{M}(P)|_Q^{*_\gamma} = P \setminus \mathcal{M}(P)|_Q + (\bigcup \mathcal{M}(P)|_Q *_\gamma Q) = ((P \setminus \mathcal{M}(P)|_Q) \cup \bigcup \mathcal{M}(P)|_Q) *_\gamma Q = P *_\gamma Q$ for some γ over $\mathbb{P}_Q$.

We now show that $Z' \subseteq Z$. Assume that each revision operation in the following is the most restrictive type, that is, for any set M, $\gamma(M) = M$. Thus, if $r \in \bigcap \gamma(\mathbb{P}_Q)$, then $r \in R$ for all $R \in \mathbb{P}_Q$. For each $\mathbb{M}$ as specified in Line 3 of ModChange, let z' be the set of rules eliminated during the revision of $\bigcup \mathbb{M}$ by Q: $z' = \bigcup \mathbb{M} \setminus (\bigcup \mathbb{M} *_\gamma Q)$. From $SE(P) \subseteq SE(\bigcup \mathbb{M})$ it then follows that $z' \cap \bigcap \gamma(\mathbb{P}_Q) = \emptyset$. Consequently, $Z' \cap \bigcap \gamma(\mathbb{P}_Q) = \emptyset$.

Analogous for contraction. □

C

Proofs for Chapter 6

Lemma 6.3: Let $P, Q, R \in \mathcal{LP}_{\mathcal{A}}$ and $\preceq$ be an ensconcement associated with P.

a) If $P + Q$ is satisfiable, then $cut_{\preceq}(Q) = P$.

b) If Q is satisfiable, then $cut_{\preceq}(Q) + Q$ is satisfiable.

c) If Q is not satisfiable, then $cut_{\preceq}(Q) = \emptyset$.

d) If $Q \models_s R$, then $cut_{\preceq}(Q) \subseteq cut_{\preceq}(R)$.

e) $cut_{\preceq}(Q + R) \subseteq cut_{\preceq}(Q)$.

f) If $cut_{\preceq}(Q) \models_s R$, then $cut_{\preceq}(Q + R) = cut_{\preceq}(Q)$.

Proof. a) Follows directly from Definition 6.2.

b) Follows directly from Definition 6.2.

c) Follows directly from Definition 6.2.

d) Follows directly from Definition 6.2.

e) Follows directly from d).

f) Let $cut_{\preceq}(Q) \models_s R$. It follows that $SE(cut_{\preceq}(Q)) \cap SE(Q) \subseteq SE(R)$, which implies $SE(cut_{\preceq}(Q)) \cap SE(Q) \cap SE(R) \neq \emptyset$ since $SE(cut_{\preceq}(Q)) \cap SE(Q) \neq \emptyset$ by Definition 6.2. We can rewrite this as $SE(cut_{\preceq}(Q)) \cap SE(Q + R) \neq \emptyset$. Thus, $cut_{\preceq}(Q) \subseteq cut_{\preceq}(Q + R)$ by Definition 6.2. By Lemma 6.3 e), we obtain $cut_{\preceq}(Q) = cut_{\preceq}(Q + R)$.

□

Proposition 6.6: Let $P, Q \in \mathcal{LP}_{\mathcal{A}}$ and $\preceq$ be an ensconcement associated with P. Then $P *_{\preceq} Q \equiv_s cut_{\preceq}(Q) + Q$.

Proof. Let $P, Q \in \mathcal{LP}_{\mathcal{A}}$ and $\preceq$ be an ensconcement associated with P. If Q is not satisfiable, then by Definition 6.4 we have $SE(P *_{\preceq} Q) = SE(P + Q) = \emptyset$ and by Lemma 6.3 c) we also have $SE(cut_{\preceq}(Q)+Q) = SE(Q) = \emptyset$. Otherwise, by Definition 6.4, $P *_{\preceq} Q = cut_{\preceq}(Q) \cup (P *_{\preceq} Q) \setminus (cut_{\preceq}(Q) + Q) \cup Q$, which means $SE(P *_{\preceq} Q) = SE(cut_{\preceq}(Q)) \cap SE((P *_{\preceq} Q) \setminus (cut_{\preceq}(Q) + Q)) \cap SE(Q)$. Since for any $r \in (P *_{\preceq} Q) \setminus (cut_{\preceq}(Q) + Q) : SE(cut_{\preceq}(Q)) \cap SE(Q) \subseteq SE(r)$, we obtain $SE(P *_{\preceq} Q) = SE(cut_{\preceq}(Q)) \cap SE(Q)$. □

Theorem 6.7: The revision operator $*_{\preceq}$ satisfies $(*1)$–$(*6)$ and $(*8)$.

Proof. $(*1)$: Follows directly from Definition 6.4.
$(*2)$: Follows directly from Definition 6.4.
$(*3)$: If Q is not satisfiable, then $P *_{\preceq} Q = P + Q$ by Definition 6.4. Otherwise, for any $r \in P *_{\preceq} Q$ it holds that $r \in P \cup Q$, which implies $P *_{\preceq} Q \subseteq P + Q$.
$(*4)$: If $P + Q$ is satisfiable, then $cut_{\preceq}(Q) = P$ by Lemma 6.3 a). Since $SE(P) \subseteq SE(r)$ for all $r \in P$, it follows from Definition 6.4 that $P *_{\preceq} Q = P + Q$.
$(*5)$: If Q is not satisfiable, then $P *_{\preceq} Q = P + Q$ is not satisfiable. Now assume Q is satisfiable. By Lemma 6.3 b) it holds that $cut_{\preceq}(Q) + Q$ is satisfiable. Since $P *_{\preceq} Q \equiv_s cut_{\preceq}(Q) + Q$ by Proposition 6.6, it follows that $P *_{\preceq} Q$ is satisfiable.
$(*6)$: Follows directly from Definition 6.4.
$(*8)$: Let $(P *_{\preceq} Q) + R$ be satisfiable. By Definition 6.4, this means $SE(cut_{\preceq}(Q)) \cap SE((P *_{\preceq} Q) \setminus cut_{\preceq}(Q)) \cap SE(Q) \cap SE(R) \neq \emptyset$ and thus $SE(cut_{\preceq}(Q)) \cap SE(Q) \cap SE(R) \neq \emptyset$. From Lemma 6.3 e) and

Definition 6.2 it follows that $cut_{\preceq}(Q + R) = cut_{\preceq}(Q)$. Then obviously $SE(cut_{\preceq}(Q + R)) \cap SE(Q) = SE(cut_{\preceq}(Q)) \cap SE(Q)$, which implies $SE(cut_{\preceq}(Q + R)) \cap SE(Q) \cap SE(R) \subseteq SE(cut_{\preceq}(Q)) \cap SE(Q)$. It thus also holds that $\{ r \in P \mid SE(cut_{\preceq}(Q)) \cap SE(Q) \subseteq SE(r) \} \subseteq \{ r \in P \mid SE(cut_{\preceq}(Q + R)) \cap SE(Q + R) \subseteq SE(r) \}$, from which we can conclude that $(P *_{\preceq} Q) + R \subseteq P *_{\preceq} (Q + R)$. □

Theorem 6.9: The revision operator $*_{\preceq}$ satisfies (∗1b), (∗2b), and (∗5b).

Proof. (∗1b): Since (∗1b) = (∗2) and $*_{\preceq}$ satisfies (∗2), $*_{\preceq}$ also satisfies (∗1b).
(∗2b): Since (∗2b) = (∗3) and $*_{\preceq}$ satisfies (∗3), $*_{\preceq}$ also satisfies (∗2b).
(∗5b): If Q is satisfiable, then by Lemma 6.3 b) it holds that $cut_{\preceq}(Q) + Q$ is satisfiable. Since $P *_{\preceq} Q \equiv_s cut_{\preceq}(Q) + Q$ by Proposition 6.6, it follows that $P *_{\preceq} Q$ is satisfiable. □

Lemma 6.12: Let $P, Q, R \in \mathcal{LP}_{\mathcal{A}}$ and $\preceq$ be an ensconcement associated with P.

a) If $P \not\models_s Q$, then $cut^{-}_{\preceq}(Q) = P$.

b) If $\not\models_s Q$, then $cut^{-}_{\preceq}(Q) \not\models_s Q$.

c) If $\models_s Q$, then $cut^{-}_{\preceq}(Q) = \emptyset$.

d) If $Q \models_s R$, then $cut^{-}_{\preceq}(R) \subseteq cut^{-}_{\preceq}(Q)$.

e) $cut^{-}_{\preceq}(Q) \subseteq cut^{-}_{\preceq}(Q + R)$.

f) If $cut^{-}_{\preceq}(Q) \models_s R$, then $cut^{-}_{\preceq}(Q + R) = cut^{-}_{\preceq}(Q)$.

g) If $cut^{-}_{\preceq}(Q) \not\models_s R$, then $cut^{-}_{\preceq}(Q + R) = cut^{-}_{\preceq}(R)$.

Proof. a) Follows directly from the definition of $cut^{-}_{\preceq}(Q)$.

b) Follows directly from the definition of $cut^{-}_{\preceq}(Q)$.

c) Follows directly from the definition of $cut^{-}_{\preceq}(Q)$.

d) Follows directly from the definition of $cut^{-}_{\preceq}(Q)$.

e) Follows directly from d).

f) Assume $cut^{-}_{\preceq}(Q + R) \neq cut^{-}_{\preceq}(Q)$. Then $cut^{-}_{\preceq}(Q) \subset cut^{-}_{\preceq}(Q + R)$ by Lemma 6.12 e). Let $r \in cut^{-}_{\preceq}(Q + R) \setminus cut^{-}_{\preceq}(Q)$. This means $SE(\{\, r' \in P \mid r \preceq r' \,\}) \cap \overline{SE(Q + R)} \neq \emptyset$ and $SE(\{\, r' \in P \mid r \preceq r' \,\}) \cap \overline{SE(Q)} = \emptyset$ by Definition of $cut^{-}_{\preceq}(Q)$. Thus, $SE(\{\, r' \in P \mid r \preceq r' \,\}) \cap \overline{SE(R)} \neq \emptyset$. Furthermore, $cut^{-}_{\preceq}(Q) \subseteq \{\, r' \in P \mid r \preceq r' \,\}$ by Definition of $cut^{-}_{\preceq}(Q)$. We therefore obtain $SE(cut^{-}_{\preceq}(Q)) \cap \overline{SE(R)} \neq \emptyset$, which implies $cut^{-}_{\preceq}(Q) \not\models_s R$.

g) Assume $cut^{-}_{\preceq}(Q + R) \neq cut^{-}_{\preceq}(R)$. Then $cut^{-}_{\preceq}(R) \subset cut^{-}_{\preceq}(Q + R)$ by Lemma 6.12 e). Let $r \in cut^{-}_{\preceq}(Q + R) \setminus cut^{-}_{\preceq}(R)$. This means $SE(\{\, r' \in P \mid r \preceq r' \,\}) \cap \overline{SE(Q + R)} \neq \emptyset$ and $SE(\{\, r' \in P \mid r \preceq r' \,\}) \cap \overline{SE(R)} = \emptyset$ by Definition of $cut^{-}_{\preceq}(R)$. Thus, $SE(\{\, r' \in P \mid r \preceq r' \,\}) \cap \overline{SE(Q)} \neq \emptyset$ and so $\{\, r' \in P \mid r \preceq r' \,\} \subseteq cut^{-}_{\preceq}(Q)$ by Definition of $cut^{-}_{\preceq}(Q)$. We therefore obtain $SE(cut^{-}_{\preceq}(Q)) \cap \overline{SE(R)} = \emptyset$, which implies $cut^{-}_{\preceq}(Q) \models_s R$.

□

Proposition 6.14: Let $P, Q \in \mathcal{LP}_{\mathcal{A}}$ and $\preceq$ be an ensconcement associated with P. Then $SE(P \dot{-}_{\preceq} Q) \cap \overline{SE(Q)} = SE(cut^{-}_{\preceq}(Q)) \cap \overline{SE(Q)}$.

Proof. If $\models_s Q$, then $\overline{SE(Q)} = \emptyset$ and $SE(P \dot{-}_{\preceq} Q) \cap \overline{SE(Q)} = SE(cut^{-}_{\preceq}(Q)) \cap \overline{SE(Q)}$. Otherwise, by Definition 6.13, $P \dot{-}_{\preceq} Q = cut^{-}_{\preceq}(Q) \cup (P \dot{-}_{\preceq} Q) \setminus cut^{-}_{\preceq}(Q)$, which means $SE(P \dot{-}_{\preceq} Q) \cap \overline{SE(Q)} = SE(cut^{-}_{\preceq}(Q)) \cap SE((P \dot{-}_{\preceq} Q) \setminus cut^{-}_{\preceq}(Q)) \cap \overline{SE(Q)}$. For all $r \in (P \dot{-}_{\preceq} Q) \setminus cut^{-}_{\preceq}(Q)$ we have $SE(cut^{-}_{\preceq}(Q)) \cap \overline{SE(Q)} \subseteq SE(r)$, so that we obtain $SE(P \dot{-}_{\preceq} Q) \cap \overline{SE(Q)} = SE(cut^{-}_{\preceq}(Q)) \cap \overline{SE(Q)}$. □

Theorem 6.15: The contraction operator $\dot{-}_{\preceq}$ satisfies ($\dot{-}$1)–($\dot{-}$4) and ($\dot{-}$6)–($\dot{-}$8).

Proof. ($\dot{-}$1): Follows directly from Definition 6.13.
($\dot{-}$2): Follows directly from Definition 6.13.
($\dot{-}$3): If $P \not\models_s Q$, then $cut^{-}_{\preceq}(Q) = P$ by Lemma 6.12 a). Since $SE(P) \subseteq SE(r)$ for all $r \in P$, this means $SE(cut^{-}_{\preceq}(Q)) \cap \overline{SE(Q)} \subseteq SE(r)$ for all $r \in P$ and by Definition 6.13 we thus have $P \dot{-}_{\preceq} Q = P$.
($\dot{-}$4): Let $\not\models_s Q$. If $P \not\models_s Q$, then $P \dot{-}_{\preceq} Q = P \not\models_s Q$ by ($\dot{-}$3). Now assume $P \models_s Q$ and let $S = SE(cut^{-}_{\preceq}(Q)) \cap \overline{SE(Q)}$. For each $r \in P$, if $r \in P \dot{-}_{\preceq} Q$, then $S \subseteq SE(r)$ by Definition 6.13, and thus $S \subseteq SE(P \dot{-}_{\preceq} Q)$. Since $S \cap SE(Q) = \emptyset$, we obtain $P \dot{-}_{\preceq} Q \not\models_s Q$.
($\dot{-}$6): Follows directly from Definition 6.13.

($\dot{-}7$): For all $r \in (P \dot{-}_{\preceq} Q) \cap (P \dot{-}_{\preceq} R) : SE(cut^{-}_{\preceq}(Q)) \cap \overline{SE(Q)} \subseteq SE(r)$ and $SE(cut^{-}_{\preceq}(R)) \cap \overline{SE(R)} \subseteq SE(r)$. This implies $SE(cut^{-}_{\preceq}(Q+R)) \cap \overline{SE(Q)} \subseteq SE(r)$ since $cut^{-}_{\preceq}(Q) \subseteq cut^{-}_{\preceq}(Q+R)$ by Lemma 6.12 e) and $SE(cut^{-}_{\preceq}(Q+R)) \cap \overline{SE(R)} \subseteq SE(r)$ since $cut^{-}_{\preceq}(R) \subseteq cut^{-}_{\preceq}(Q+R)$ by Lemma 6.12 e). From $\overline{SE(Q+R)} = \overline{SE(Q)} \cup \overline{SE(R)}$ we obtain $SE(cut^{-}_{\preceq}(Q+R)) \cap \overline{SE(Q+R)} \subseteq SE(r)$ and thus $r \in P \dot{-}_{\preceq} (Q+R)$ by Definition 6.13.
($\dot{-}8$): Assume $SE(P \dot{-}_{\preceq} (Q+R)) \not\subseteq SE(Q)$. Then, $SE(cut^{-}_{\preceq}(Q+R)) \cap SE((P \dot{-}_{\preceq} (Q+R)) \setminus cut^{-}_{\preceq}(Q+R)) \not\subseteq SE(Q)$, which means $SE(cut^{-}_{\preceq}(Q+R)) \cap \overline{SE(Q)} \neq \emptyset$ (i). Recall that $cut^{-}_{\preceq}(Q)$ is maximal and $cut^{-}_{\preceq}(Q) \subseteq cut^{-}_{\preceq}(Q+R)$ (ii) by Lemma 6.12 e). From (i) and (ii) it follows that $cut^{-}_{\preceq}(Q) = cut^{-}_{\preceq}(Q+R)$. Since $\overline{SE(Q)} \subseteq \overline{SE(Q+R)}$, we have $SE(cut^{-}_{\preceq}(Q)) \cap \overline{SE(Q)} \subseteq SE(cut^{-}_{\preceq}(Q+R)) \cap \overline{SE(Q+R)}$, which implies $P \dot{-}_{\preceq} (Q+R) \subseteq P \dot{-}_{\preceq} Q$ by Definition 6.13. □

Theorem 6.17: The contraction operator $\dot{-}_{\preceq}$ satisfies (−1b), (−2b) and (−5b)–(−8b).

Proof. ($\dot{-}$1b): Follows from ($\dot{-}$1b) = ($\dot{-}$2) and satisfaction of ($\dot{-}$2).
($\dot{-}$2b): Follows from ($\dot{-}$2b) = ($\dot{-}$4) and satisfaction of ($\dot{-}$4).
($\dot{-}$5b): Follows from ($\dot{-}$5b) = ($\dot{-}$3) and satisfaction of ($\dot{-}$3).
($\dot{-}$6b): Follows from ($\dot{-}$6b) = ($\dot{-}$6) and satisfaction of ($\dot{-}$6).
($\dot{-}$7b): If $\models_s Q+R$, then $P \dot{-}_{\preceq} (Q+R) = P$ by Definition 6.13 and $\models_s Q$ and $\models_s R$, which means $P \dot{-}_{\preceq} Q = P$ and $P \dot{-}_{\preceq} R = P$ by Definition 6.13. Now let $\not\models_s Q+R$. We proceed by cases.

Case 1: $cut^{-}_{\preceq}(Q) \models_s R$. Then $cut^{-}_{\preceq}(Q+R) = cut^{-}_{\preceq}(Q)$ by Lemma 6.12 f). Let $r \in P \dot{-}_{\preceq} (Q+R)$. This means $r \in P$ by ($\dot{-}$2) and $SE(cut^{-}_{\preceq}(Q+R)) \cap \overline{SE(Q+R)} \subseteq SE(r)$ by Definition 6.13. It follows that $SE(cut^{-}_{\preceq}(Q+R)) \cap \overline{SE(Q)} \subseteq SE(r)$. Due to the case assumption, we obtain $SE(cut^{-}_{\preceq}(Q)) \cap \overline{SE(Q)} \subseteq SE(r)$ and thus $r \in P \dot{-}_{\preceq} Q$ by Definition 6.13. Now let $r \in P \dot{-}_{\preceq} Q$. This means $r \in P$ by ($\dot{-}$2) and $SE(cut^{-}_{\preceq}(Q)) \cap \overline{SE(Q)} \subseteq SE(r)$ by Definition 6.13. Then $SE(cut^{-}_{\preceq}(Q+R)) \cap \overline{SE(Q)} \subseteq SE(r)$ due to the case assumption. It also follows from the case assumption that $SE(cut^{-}_{\preceq}(Q)) \cap \overline{SE(R)} = \emptyset$ and $SE(cut^{-}_{\preceq}(Q+R)) \cap \overline{SE(R)} = \emptyset$. We thus have $SE(cut^{-}_{\preceq}(Q+R)) \cap (\overline{SE(Q)} \cup \overline{SE(R)}) \subseteq SE(r)$, that is, $SE(cut^{-}_{\preceq}(Q+R)) \cap \overline{SE(Q+R)} \subseteq SE(r)$. Therefore, $r \in P \dot{-}_{\preceq} (Q+R)$ by Definition 6.13.

Case 2: $cut^{-}_{\preceq}(R) \models_s Q$. Follows analogous to Case 1 so that $P \dot{-}_{\preceq} (Q+R) = P \dot{-}_{\preceq} R$.

Case 3: $cut^{-}_{\preceq}(Q) \not\models_s R$ and $cut^{-}_{\preceq}(R) \not\models_s Q$. Then $cut^{-}_{\preceq}(Q+R) = cut^{-}_{\preceq}(Q) = cut^{-}_{\preceq}(R)$ by Lemma 6.12 g). Let $r \in P \dot{-}_{\preceq} (Q+R)$. This means $r \in P$ by ($\dot{-}2$) and $SE(cut^{-}_{\preceq}(Q+R)) \cap \overline{SE(Q+R)} \subseteq SE(r)$ by Definition 6.13. We thus have $SE(cut^{-}_{\preceq}(Q+R)) \cap \overline{SE(Q)} \subseteq SE(r)$ and $SE(cut^{-}_{\preceq}(Q+R)) \cap \overline{SE(R)} \subseteq SE(r)$. From the case assumption it follows that $SE(cut^{-}_{\preceq}(Q)) \cap \overline{SE(Q)} \subseteq SE(r)$ and $SE(cut^{-}_{\preceq}(R)) \cap \overline{SE(R)} \subseteq SE(r)$. This means $r \in P \dot{-}_{\preceq} Q$ and $r \in P \dot{-}_{\preceq} R$ by Definition 6.13 and therefore $r \in (P \dot{-}_{\preceq} Q) \cap (P \dot{-}_{\preceq} R)$.
($\dot{-}$8b): Assume $SE(P \dot{-}_{\preceq} Q) \subseteq SE(Q) \cup SE(r)$, that is, $SE(P \dot{-}_{\preceq} Q) \cap \overline{SE(Q)} \subseteq SE(r)$. By Proposition 6.14, $SE(P \dot{-}_{\preceq} Q) \cap \overline{SE(Q)} = SE(cut^{-}_{\preceq}(Q)) \cap \overline{SE(Q)}$, so that we obtain $SE(cut^{-}_{\preceq}(Q)) \cap \overline{SE(Q)} \subseteq SE(r)$. This implies $r \in P \dot{-}_{\preceq} Q$ by Definition 6.13. We can conclude $r \in P$ by ($\dot{-}2$). □

Proposition 6.20: Let $P \in \mathcal{LP}_{\mathcal{A}}$, $\preceq$ be an ensconcement associated with P, and $*$ an operator for P such that for any $Q \in \mathcal{LP}_{\mathcal{A}}$: $P * Q = (P \dot{-}_{\preceq} \overline{Q}) + Q$. Then $P * Q = P *_{\preceq} Q$.

Proof. Let $\overline{SE(Q)} = \mathcal{SE}$. Then $P \dot{-}_{\preceq} \overline{Q} = P$ by Definition 6.13 and thus $P * Q = (P \dot{-}_{\preceq} \overline{Q}) + Q = P + Q = P *_{\preceq} Q$ by Definition 6.4. Otherwise, $\overline{SE(Q)} \neq \mathcal{SE}$ such that $cut^{-}_{\preceq}(\overline{Q}) = \{\, r \in P \mid SE(\{\, r' \in P \mid r \preceq r' \,\}) \cap SE(Q) \neq \emptyset \,\} = cut_{\preceq}(Q)$ by Definition 6.2. It follows that $P * Q = (P \dot{-}_{\preceq} \overline{Q}) + Q = \{\, r \in P \mid SE(cut_{\preceq}(Q)) \cap SE(Q) \subseteq SE(r) \,\} + Q = P *_{\preceq} Q$ by Definitions 6.4 and 6.13. □

Proposition 6.21: Let $P \in \mathcal{LP}_{\mathcal{A}}$, $\preceq$ be an ensconcement associated with P, and $\dot{-}$ an operator for P such that for any $Q \in \mathcal{LP}_{\mathcal{A}}$: $P \dot{-} Q = P \cap (P *_{\preceq} \overline{Q})$. Then $P \dot{-} Q = P \dot{-}_{\preceq} Q$.

Proof. Let $\overline{SE(Q)} = \emptyset$. Then $P *_{\preceq} \overline{Q} = P + \overline{Q}$ by Definition 6.4 and thus $P \dot{-} Q = P \cap (P *_{\preceq} \overline{Q}) = P = P \dot{-}_{\preceq} Q$ by Definition 6.13. Otherwise, $\overline{SE(Q)} \neq \emptyset$ such that $cut_{\preceq}(\overline{Q}) = \{\, r \in P \mid SE(\{\, r' \in P \mid r \preceq r' \,\}) \cap \overline{SE(Q)} \neq \emptyset \,\} = cut^{-}_{\preceq}(Q)$ by definition of $cut^{-}_{\preceq}(Q)$. It follows that $P \dot{-} Q = P \cap (P *_{\preceq} \overline{Q}) = P \cap (\{\, r \in P \mid SE(cut^{-}_{\preceq}(Q)) \cap \overline{SE(Q)} \subseteq SE(r) \,\} + \overline{Q})$ by Definition 6.4. Assume there exists a rule $r' \in \overline{Q}$ with $r' \in P \setminus \{\, r \in P \mid SE(cut^{-}_{\preceq}(Q)) \cap \overline{SE(Q)} \subseteq SE(r) \,\}$. Since $SE(\overline{Q}) = \overline{SE(Q)} \subseteq SE(r')$, it holds that $SE(cut^{-}_{\preceq}(Q)) \cap \overline{SE(Q)} \subseteq SE(r')$. This implies $r' \in \{\, r \in P \mid SE(cut^{-}_{\preceq}(Q)) \cap \overline{SE(Q)} \subseteq SE(r) \,\}$, a contradiction. We therefore obtain $P \dot{-} Q = P \dot{-}_{\preceq} Q$ by Definition 6.13. □

Theorem 6.22: Let $P, Q \in \mathcal{LP}_{\mathcal{A}}$. For any selection function γ, there exists an ensconcement $\preceq$ associated with P such that $P *_{\gamma} Q = P *_{\preceq} Q$.

Proof. Let $P, Q \in \mathcal{LP}_{\mathcal{A}}$ and γ be a selection function that determines the outcome of $P *_{\gamma} Q$. By $S = (P *_{\gamma} Q) \cap P = \bigcap \gamma(\mathbb{P}_Q)$ we denote the subset of P that is retained in the revision and by $S' = P \setminus S$ the subset of P that is discarded. We can then create an ensconcement $\preceq$ associated with P that has a minimal number of levels, such that for all $r \in S$ and for all $r' \in S'$: $r' \prec r$. We now show that $(P *_{\preceq} Q) \cap P = S$. Clearly, $cut_{\preceq}(Q) = S$ by Definition 6.2, which implies $S \subseteq (P *_{\preceq} Q) \cap P$. Assume that there exists an $r' \in S'$ with $SE(cut_{\preceq}(Q)) \cap SE(Q) \subseteq SE(r')$. Then for each selected compatible set $R \in \gamma(\mathbb{P}_Q)$ it would hold that $r' \in R$ because R is maximal by the definition of $\mathbb{P}_Q$. Yet this implies $r' \in S$, a contradiction. □

Theorem 6.24: Let $P, Q \in \mathcal{LP}_{\mathcal{A}}$. For any selection function γ, there exists an ensconcement $\preceq$ associated with P such that $P \dot{-}_{\gamma} Q = P \dot{-}_{\preceq} Q$.

Proof. Follows analogously to the proof of Theorem 6.22. □

Lemma 6.28: Let $\preceq^R$ be a subset-ensconcement associated with some $P \in \mathcal{LP}_{\mathcal{A}}$ and $R \subseteq P$. For any rule $r \in R$, it holds that $R \preceq^R \{r\}$.

Proof. Since $R \models_s \{r\}$, it follows from Conditions ($\preceq^R$1) and ($\preceq^R$2) that $\{r\} \not\preceq^R R$. □

Theorem 6.29: Let $P, Q \in \mathcal{LP}_{\mathcal{A}}$, $\preceq$ be an ensconcement associated with P, and $\preceq^R$ a subset-ensconcement associated with P such that $\{r\} \preceq^R \{r'\}$ iff $r \preceq r'$ for all $r, r' \in P$. Then $P *_{\preceq} Q = P *_{\preceq^R} Q$ (or $P \dot{-}_{\preceq} Q = P \dot{-}_{\preceq^R} Q$, alternatively).

Proof. Let $P, Q \in \mathcal{LP}_{\mathcal{A}}$, $\preceq$ an ensconcement associated with P, and $\preceq^R$ a subset-ensconcement associated with P. Assume that $\{r\} \preceq^R \{r'\}$ iff $r \preceq r'$ for all $r, r' \in P$. From Lemma 6.28 it is clear that $cut_{\preceq}(Q) = cut_{\preceq^R}(Q)$, which implies for all $r \in (P *_{\preceq} Q) \setminus Q : SE(cut_{\preceq^R}(Q)) \cap SE(Q) \subseteq SE(\{r\})$, and thus $P *_{\preceq} Q \subseteq P *_{\preceq^R} Q$. Since $SE(R) \subseteq SE(r)$ for any $R \subseteq P$ and each $r \in R$, we also have $P *_{\preceq^R} Q \subseteq P *_{\preceq} Q$.

Analogous for contraction. □

Corollary 6.30: For any $P, Q \in \mathcal{LP}_{\mathcal{A}}$, let $P \setminus \mathcal{M}(P)|_Q = \{\, r \in P \mid \forall M \in \mathcal{M}(P)|_Q : r \notin M \,\}$ and $\mathcal{M}(P)|_Q^{\circ}$ denote the output of Algorithm 1 for the inputs $\mathcal{M}(P)|_Q$, $\circ \in \{*_{\preceq}, \dot{-}_{\preceq}\}$, and Q. Then $P *_{\preceq} Q = P \setminus \mathcal{M}(P)|_Q + \bigcup \mathcal{M}(P)|_Q^{*_{\preceq}} + Q$ (or $P \dot{-}_{\preceq} Q = P \setminus \mathcal{M}(P)|_Q + \bigcup \mathcal{M}(P)|_Q^{\dot{-}_{\preceq}}$, respectively) for some ensconcement $\preceq$ associated with P.

Proof. Follows directly from Theorems 5.28, 6.22, and 6.24. □

Theorem 6.36: The contraction operator $\dot{-}_{\leq_Q}$ satisfies ($\dot{-}1$)–($\dot{-}4$) and ($\dot{-}6$).

Proof. ($\dot{-}1$): Follows directly from Definition 6.34.
($\dot{-}2$): Follows directly from Definition 6.34.
($\dot{-}3$): Suppose $P \not\models_s Q$. By ($\dot{-}2$) we have $P \dot{-}_{\leq_Q} Q \subseteq P$ and need to show $P \subseteq P \dot{-}_{\leq_Q} Q$. Let $R \subseteq P$. Since $SE(R) \subseteq SE(Q) \cup SE(R)$ it follows from Definition 6.33 that $R \in C_Q(R)$. Due to the assumption and ($\leq_Q 4$) we obtain $Q <_Q R$. Thus, we can conclude $R \subseteq P \dot{-}_{\leq_Q} Q$ by Definition 6.34.
($\dot{-}4$): Let $\not\models_s Q$. If $P \not\models_s Q$, then $P \dot{-}_{\leq_Q} Q = P \not\models_s Q$ by ($\dot{-}3$). Now assume $P \models_s Q$ and let $R \subseteq P$ such that $R \not\models_s Q$. For all $R' \in C_Q(R)$, $SE(R') \subseteq SE(Q) \cup SE(R)$ implies $SE(R') \subseteq SE(Q)$ according to the assumption and thus $R' \leq_Q Q$ by ($\leq_Q 2$). By Definition 6.34, we then obtain $R \nsubseteq P \dot{-}_{\leq_Q} Q$.
($\dot{-}6$): Follows directly from Definition 6.34. □

Theorem 6.38: The revision operator $*_{\leq_Q}$ satisfies ($*1$)–($*6$).

Proof. ($*1$): Follows directly from Definition 6.37.
($*2$): Follows directly from Definition 6.37.
($*3$): Follows directly from Definition 6.37.
($*4$): Let $P + Q$ be satisfiable. This means $SE(P) \cap SE(Q) \neq \emptyset$. Thus $SE(P) \nsubseteq \overline{SE(Q)}$, that is, $P \not\models_s \overline{Q}$. By ($\dot{-}3$) we obtain $P \dot{-}_{\leq_Q} \overline{Q} = P$ and can conclude $P *_{\leq_Q} Q = P + Q$.
($*5$): If Q is not satisfiable, then $P *_{\leq_Q} Q = (P \dot{-}_{\leq_Q} Q) + Q$ is not satisfiable since $SE(P \dot{-}_{\leq_Q} Q) \cap SE(Q) \subseteq SE(Q)$. Otherwise, $SE(Q) \neq \emptyset$. This implies $\overline{SE(Q)} \neq \mathcal{SE}$, that is, $\not\models_s \overline{Q}$. It follows from ($\dot{-}4$) that $P \dot{-}_{\leq_Q} \overline{Q} \not\models_s \overline{Q}$, that is, $SE(P \dot{-}_{\leq_Q} \overline{Q}) \nsubseteq \overline{SE(Q)}$. Therefore, $SE(P \dot{-}_{\leq_Q} \overline{Q}) \cap SE(Q) = SE(P *_{\leq_Q} Q) \neq \emptyset$.
($*6$): Follows directly from Definition 6.37. □

D

Proofs for Chapter 8

Proposition 8.3: Let $P, Q \in \mathcal{LP}_{\mathcal{A}}$. For any maxichoice selection function γ_P for P, there exists a selection function γ for P such that $(P \cup Q)!_{\gamma_P}^{SE} Q = P *_{\gamma} Q$.

Proof. Let $P, Q \in \mathcal{LP}_{\mathcal{A}}$ and γ_P be a maxichoice selection function for P. We will prove by cases.

Case 1: Q is not satisfiable. This means $(P \cup Q)\perp_!^{SE} Q = \emptyset$ by definition of $P\perp_!^{SE} Q$. Then $\gamma_P((P \cup Q)\perp_!^{SE} Q) = P \cup Q$ by definition of γ_P. It follows that $(P \cup Q)!_{\gamma_P}^{SE} Q = P \cup Q = P *_{\gamma} Q$ by definition of $!_{\gamma_P}^{SE}$ and Definition 5.3.

Case 2: Q is satisfiable and for all $R \subseteq P : SE(R \cup Q) = \emptyset$. Then $(P \cup Q)\perp_!^{SE} Q = \{Q\}$ by definition of $\perp_!^{SE}$ and $\mathbb{P}_Q = \emptyset = \bigcap \gamma(\mathbb{P}_Q)$ by Definition 5.1. It follows that $(P \cup Q)!_{\gamma_P}^{SE} Q = \gamma_P((P \cup Q)\perp_!^{SE} Q) = Q = P *_{\gamma} Q$ by definition of $!_{\gamma_P}^{SE}$ and Definition 5.3.

Case 3: Q is satisfiable and there exists $R \subseteq P : SE(R \cup Q) \neq \emptyset$ such that for all R' with $R \subset R' \subseteq P : SE(R') = \emptyset$. Then $R \cup Q \in (P \cup Q)\perp_!^{SE} Q$ by definition of $\perp_!^{SE}$ and $R \in \mathbb{P}_Q$ by Definition 5.1. Let γ be a maxichoice selection function for P such that $\gamma(2^P) \cup Q = \gamma_P(\{\, R \cup Q \mid R \in 2^P \,\})$. This implies $\gamma_P((P \cup Q)\perp_!^{SE} Q) = \gamma(\mathbb{P}_Q) \cup Q = \bigcap \gamma(\mathbb{P}_Q) \cup Q$. Thus, $(P \cup Q)!_{\gamma_P}^{SE} Q = P *_{\gamma} Q$ by definition of $!_{\gamma_P}^{SE}$ and Definition 5.3. □

Proposition 8.6: Let $P, Q \in \mathcal{LP}_{\mathcal{A}}$ and γ^1, γ_P be single-choice and maxichoice selection functions, respectively, for P. If $\gamma^1(2^P) \cup Q = \gamma_P(\{\, R \cup Q \mid R \in 2^P \,\})$ for any γ_P, then $P *^{AS}_{\gamma^1} Q = (P \cup Q)!_{\gamma_P} Q$.

Proof. Let $P, Q \in \mathcal{LP}_{\mathcal{A}}$ and γ^1, γ_P be single-choice and maxichoice selection functions, respectively, for P. Assume that $\gamma^1(2^P) \cup Q = \gamma_P(\{\, R \cup Q \mid R \in 2^P \,\})$ for any γ_P. We will prove by cases.

Case 1: $AS(Q) = \emptyset$ and for all $R \subseteq P : AS(R \cup Q) = \emptyset$. This implies $\mathbb{P}^{AS}_Q = \emptyset$ by definition of $\mathbb{P}^{AS}_Q$ and thus $P *_\gamma Q = P \cup Q$ by Definition 8.5. It also implies $(P \cup Q) \perp_! Q = \emptyset$ by definition of $P \perp_! Q$ and therefore $\gamma_P((P \cup Q) \perp_! Q) = P \cup Q$ by definition of γ_P. We can conclude $(P \cup Q)!_{\gamma_P} Q = P \cup Q$ by Definition 3.3.

Case 2: $AS(Q) \neq \emptyset$ and for all $R \subseteq P : AS(R \cup Q) = \emptyset$. Then $\mathbb{P}^{AS}_Q = \emptyset = \gamma^1(\mathbb{P}^{AS}_Q)$ by definitions of $\mathbb{P}^{AS}_Q$ and γ^1. We also have $(P \cup Q) \perp_! Q = \{Q\} = \gamma_P((P \cup Q) \perp_! Q)$ by definitions of $\perp_!$ and γ_P. It follows that $P *^{AS}_{\gamma^1} Q = Q = (P \cup Q)!_{\gamma_P} Q$ by Definitions 8.5 and 3.3.

Case 3: There exists $R \subseteq P : AS(R \cup Q) \neq \emptyset$ and for all R' with $R \subset R' \subseteq P : AS(R' \cup Q) = \emptyset$. Then $R \in \mathbb{P}^{AS}_Q$ by definition of $\mathbb{P}^{AS}_Q$ and $R \cup Q \in (P \cup Q) \perp_! Q$ by definition of $\perp_!$. Due to the assumption, it holds that $\gamma^1(\mathbb{P}^{AS}_Q) \cup Q = \gamma_P((P \cup Q) \perp_! Q)$. Thus, $P *^{AS}_{\gamma^1} Q = (P \cup Q)!_{\gamma_P} Q$ by Definitions 8.5 and 3.3. □

E

Proofs for Chapter 9

Lemma 9.4: Let Π be a normal DL logic program relative to a DL knowledge base L, l a literal occurring in $ground(\Pi)$, and $I^+ \subseteq HB_\Pi$ consistent with L. Then $I^+ \models_L l$ iff $\langle I^+, \overline{I^+}\rangle \models_{3,L} l$.

Proof. Consider the following four cases:

(i) $l = A$ where A is a pure atom.
$I^+ \models_L A$
$\Leftrightarrow A \in I^+$
$\Leftrightarrow \langle I^+, \overline{I^+}\rangle \models_{3,L} A$.

(ii) $l = not\, A$ where A is a pure atom.
$I^+ \models_L not\, A$
$\Leftrightarrow I^+ \not\models_L A$
$\Leftrightarrow A \notin I^+$
$\Leftrightarrow A \in \overline{I^+}$
$\Leftrightarrow \langle I^+, \overline{I^+}\rangle \models_{3,L} not\, A$.

(iii) $l = A$ where A is a ground DL axiom.
$I^+ \models_L A$

$\Leftrightarrow L \cup I^+|_\Omega \cup \neg\overline{I^+}|_\Omega \models A$
$\Leftrightarrow \langle I^+, \overline{I^+}\rangle \models_{3,L} A.$

(iv) $l = not\ A$ where A is a ground DL axiom.
$I^+ \models_L not\ A$
$\Leftrightarrow I^+ \not\models_L A$
$\Leftrightarrow L \cup I^+|_\Omega \cup \neg\overline{I^+}|_\Omega \not\models A$
$\Leftrightarrow L \cup I^+|_\Omega \cup \neg\overline{I^+}|_\Omega \models \neg A$
$\Leftrightarrow \langle I^+, \overline{I^+}\rangle \models_{3,L} not\ A.$

□

Theorem 9.5: Let Π be a normal DL logic program relative to a DL knowledge base L and $I^+ \subseteq HB_\Pi$ consistent with L. Then I^+ is a two-valued model of Π iff $\langle I^+, \overline{I^+}\rangle$ is a three-valued model of Π.

Proof. "If "– proof by contrapositive: If I^+ does not satisfy all rules in $ground(\Pi)$, then there exists a rule $r \in ground(\Pi)$ such that $I^+ \not\models_L head(r)$ and $I^+ \models_L l$ for every $l \in body(r)$. By Lemma 9.4, we obtain $\langle I^+, \overline{I^+}\rangle \not\models_{3,L} head(r)$ and $\langle I^+, \overline{I^+}\rangle \models_{3,L} l$ for every $l \in body(r)$. This means that $\langle I^+, \overline{I^+}\rangle$ does not satisfy r and is therefore not a three-valued model of Π.

The proof for "only if" follows analogously. □

Lemma 9.6: Let Π be a normal DL logic program relative to a DL knowledge base L, l a literal occurring in $ground(\Pi)$, and $\langle I^+, I^-\rangle$ a three-valued interpretation of Π. If I^+ is consistent with L, then $\langle I^+, I^-\rangle \models_{3,L} l$ implies $I^+ \models_L l$.

Proof. Consider the following four cases:

(i) $l = A$ where A is a pure atom.
$\langle I^+, I^-\rangle \models_{3,L} A$
$\Rightarrow A \in I^+$
$\Rightarrow I^+ \models_L A.$

(ii) $l = not\ A$ where A is a pure atom.
$\langle I^+, I^-\rangle \models_{3,L} not\ A$
$\Rightarrow A \in I^-$

$\Rightarrow A \notin I^+ \qquad (I^+ \cap I^- = \emptyset)$
$\Rightarrow I^+ \not\models_L A$
$\Rightarrow I^+ \models_L not\ A.$

(iii) $l = A$ where A is a ground DL axiom.
$\langle I^+, I^- \rangle \models_{3,L} A$
$\Rightarrow L \cup I^+|_\Omega \cup \neg I^-|_\Omega \models A$
$\Rightarrow L \cup I^+|_\Omega \cup \neg \overline{I^+}|_\Omega \models A$
$\Rightarrow I^+ \models_L A.$

(iv) $l = not\ A$ where A is a ground DL axiom.
$\langle I^+, I^- \rangle \models_{3,L} not\ A$
$\Rightarrow L \cup I^+|_\Omega \cup \neg I^-|_\Omega \models \neg A$
$\Rightarrow L \cup I^+|_\Omega \cup \neg \overline{I^+}|_\Omega \models \neg A$
$\Rightarrow L \cup I^+|_\Omega \cup \neg \overline{I^+}|_\Omega \not\models A$
$\Rightarrow I^+ \not\models_L A$
$\Rightarrow I^+ \models_L not\ A.$

□

Theorem 9.7: Let Π be a normal DL logic program relative to a DL knowledge base L and $\langle I^+, I^- \rangle$ a three-valued model of Π. If I^+ is consistent with L, then I^+ is a two-valued model of Π.

Proof. Given any rule $r \in ground(\Pi)$, $\langle I^+, I^- \rangle$ satisfies r. If $\langle I^+, I^- \rangle \models_{3,L} head(r)$, then $I^+ \models_L head(r)$ by Lemma 9.6. Otherwise, $\langle I^+, I^- \rangle \not\models_{3,L} l$ for some $l \in body(r)$, and by Lemma 9.6, $I^+ \not\models_L l$. Hence, I^+ satisfies r and is thus a two-valued model of Π. □

Lemma 9.9: Let Π be a normal DL logic program relative to a DL knowledge base L. For any two three-valued interpretations $\langle J_1^+, I^- \rangle$ and $\langle J_2^+, I^- \rangle$ of Π with $J_1^+ \subseteq J_2^+$, $\langle J_1^+, I^- \rangle \models_{3,L} l$ implies $\langle J_2^+, I^- \rangle \models_{3,L} l$ for each literal l occurring in $ground(\Pi)$.

Proof. Consider the following four cases:

(i) $l = A$ where A is a pure atom.
$\langle J_1^+, I^- \rangle \models_{3,L} A$
$\Rightarrow A \in J_1^+$

$\Rightarrow A \in J_2^+$
$\Rightarrow \langle J_2^+, I^- \rangle \models_{3,L} A$.

(ii) $l = not\ A$ where A is a pure atom.
$\langle J_1^+, I^- \rangle \models_{3,L} not\ A$
$\Rightarrow A \in I^-$
$\Rightarrow \langle J_2^+, I^- \rangle \models_{3,L} not\ A$.

(iii) $l = A$ where A is a ground DL axiom.
$\langle J_1^+, I^- \rangle \models_{3,L} A$
$\Rightarrow L \cup J_1^+|_\Omega \cup \neg I^-|_\Omega \models A$
$\Rightarrow L \cup J_2^+|_\Omega \cup \neg I^-|_\Omega \models A$
$\Rightarrow \langle J_2^+, I^- \rangle \models_{3,L} A$.

(iv) $l = not\ A$ where A is a ground DL axiom.
$\langle J_1^+, I^- \rangle \models_{3,L} not\ A$
$\Rightarrow L \cup J_1^+|_\Omega \cup \neg I^-|_\Omega \models \neg A$
$\Rightarrow L \cup J_2^+|_\Omega \cup \neg I^-|_\Omega \models \neg A$
$\Rightarrow \langle J_2^+, I^- \rangle \models_{3,L} not\ A$.

□

Proposition 9.10: Let Π be a normal DL logic program relative to a DL knowledge base L and $I^- \subseteq HB_\Pi$. For any $J_1^+ \subseteq J_2^+ \subseteq HB_\Pi$, $T_\Pi(J_1^+, I^-) \subseteq T_\Pi(J_2^+, I^-)$.

Proof. For each $A \in T_\Pi(J_1^+, I^-)$, there exists a rule $r \in ground(\Pi)$ with $head(r) = A$ and $\langle J_1^+, I^- \rangle \models_{3,L} l$ for every $l \in body(r)$. This implies $\langle J_2^+, I^- \rangle \models_{3,L} l$ by Lemma 9.9 and thus $A \in T_\Pi(J_2^+, I^-)$. □

Lemma 9.13: Let Π be a normal DL logic program relative to a DL knowledge base L and $\langle I^+, I^- \rangle$ a three-valued interpretation of Π. For any literal l occurring in $ground(\Pi)$ and any $J^+ \subseteq I^+$:

1) $\langle J^+, I^- \rangle \models_{3,L} l$ implies $(J^+, I^+) \models_L l$, and

2) $\langle J^+, \overline{I^+} \rangle \models_{3,L} l$ iff $(J^+, I^+) \models_L l$.

Proof. Consider the following four cases:

(i) $l = A$ where A is a pure atom.
$\langle J^+, I^- \rangle \models_{3,L} A$
$\Rightarrow A \in J^+$
$\Rightarrow A \in J'$ for every J' with $J^+ \subseteq J' \subseteq I^+$
$\Rightarrow (J^+, I^+) \models_L A$.

(ii) $l = not\, A$ where A is a pure atom.
$\langle J^+, I^- \rangle \models_{3,L} not\, A$
$\Rightarrow A \in I^-$
$\Rightarrow A \notin J'$ for every J' with $J^+ \subseteq J' \subseteq I^+ \qquad (I^+ \cap I^- = \emptyset)$
$\Rightarrow (J^+, I^+) \models_L not\, A$.

(iii) $l = A$ where A is a ground DL axiom.
$\langle J^+, I^- \rangle \models_{3,L} A$
$\Rightarrow L \cup J^+|_\Omega \cup \neg I^-|_\Omega \models A$
$\Rightarrow L \cup J'|_\Omega \cup \neg I^-|_\Omega \models A$ for every J' with $J^+ \subseteq J' \subseteq I^+$
$\Rightarrow L \cup J'|_\Omega \cup \neg \overline{J'}|_\Omega \models A$ for every J' with $J^+ \subseteq J' \subseteq I^+$
$\Rightarrow (J^+, I^+) \models_L A$.

(iv) $l = not\, A$ where A is a ground DL axiom.
$\langle J^+, I^- \rangle \models_{3,L} not\, A$
$\Rightarrow L \cup J^+|_\Omega \cup \neg I^-|_\Omega \models \neg A$
$\Rightarrow L \cup J'|_\Omega \cup \neg I^-|_\Omega \models \neg A$ for every J' with $J^+ \subseteq J' \subseteq I^+$
$\Rightarrow L \cup J'|_\Omega \cup \neg \overline{J'}|_\Omega \models \neg A$ for every J' with $J^+ \subseteq J' \subseteq I^+$
$\Rightarrow L \cup J'|_\Omega \cup \neg \overline{J'}|_\Omega \not\models A$ for every J' with $J^+ \subseteq J' \subseteq I^+$
$\Rightarrow (J^+, I^+) \models_L not\, A$.

The second part of the lemma can be proved analogously. □

Lemma 9.14: Let Π be a normal DL logic program relative to a DL knowledge base L, I^+ a two-valued model of Π, and $\langle I^+, I^- \rangle$ a three-valued model of Π. For every $n \geq 0$, $T^n_\Pi(\emptyset, I^-) \subseteq \mathcal{T}^n_\Pi(\emptyset, I^+)$.

Proof. We prove this by induction on n.
Base: $n = 0$. It follows directly from the definition.
Step: Suppose it holds for $n = k$. We prove it for $n = k + 1$.
$A \in T^{k+1}_\Pi(\emptyset, I^-)$
$\Rightarrow A \in T_\Pi(T^k_\Pi(\emptyset, I^-), I^-)$ by Definition 9.8
$\Rightarrow A \in T_\Pi(\mathcal{T}^k_\Pi(\emptyset, I^+), I^-)$ by the induction hypothesis $T^k_\Pi(\emptyset, I^-) \subseteq \mathcal{T}^k_\Pi(\emptyset, I^+)$,

by monotonicity of T_Π on its first argument, and by $\mathcal{T}_\Pi^k(\emptyset, I^+) \subseteq HB_\Pi$ according Theorem 4 of (Shen & Wang 2011) since I^+ is a two-valued model of Π
$\Rightarrow \exists r \in ground(\Pi)$ with $head(r) = A$ and $\langle \mathcal{T}_\Pi^k(\emptyset, I^+), I^- \rangle \models_{3,L} l$ for every $l \in body(r)$
$\Rightarrow A \in \mathcal{T}_\Pi(\mathcal{T}_\Pi^k(\emptyset, I^+), I^+)$ since $(\mathcal{T}_\Pi^k(\emptyset, I^+), I^+) \models_L l$ for every $l \in body(r)$ due to Lemma 9.13
$\Rightarrow A \in \mathcal{T}_\Pi^{k+1}(\emptyset, I^+)$. □

Theorem 9.15: Let Π be a normal DL logic program relative to a DL knowledge base L and $\langle I^+, I^- \rangle$ a three-valued answer set of Π. Then I^+ is a two-valued answer set of Π iff $\langle I^+, \overline{I^+} \rangle$ is a three-valued model of Π.

Proof. "Only if". If I^+ is a two-valued answer set of Π, this means that I^+ is a two-valued model of Π. By Theorem 9.5, it follows that $\langle I^+, \overline{I^+} \rangle$ is a three-valued model of Π.

"If". Firstly, assume that $\langle I^+, \overline{I^+} \rangle$ is a three-valued model of Π and I^+ is not a two-valued model of Π. This is a contradiction by Theorem 9.5. Secondly, we show that for any $A \in I^+$, either $A \in \mathcal{T}_\Pi^\alpha(\emptyset, I^+)$ or $L \cup \mathcal{T}_\Pi^\alpha(\emptyset, I^+)|_\Omega \cup \neg\overline{I^+}|_\Omega \models A$. Recall that $\langle I^+, I^- \rangle$ is a three-valued answer set of Π. This implies either $A \in T_\Pi^\alpha(\emptyset, I^-)$ or $L \cup T_\Pi^\alpha(\emptyset, I^-)|_\Omega \cup \neg I^-|_\Omega \models A$. It follows from Lemma 9.14 that either $A \in \mathcal{T}_\Pi^\alpha(\emptyset, I^+)$ or $L \cup \mathcal{T}_\Pi^\alpha(\emptyset, I^+)|_\Omega \cup \neg\overline{I^+}|_\Omega \models A$. □

Lemma 9.17: Let Π be a normal DL logic program relative to a DL knowledge base L, I^+ a two-valued model of Π, and $\langle I^+, \overline{I^+} \rangle$ a three-valued model of Π. For every $n \geq 0$, $T_\Pi^n(\emptyset, \overline{I^+}) = \mathcal{T}_\Pi^n(\emptyset, I^+)$.

Proof. We prove this by induction on n.
Base: $n = 0$. It follows directly from the definition.
Step: Suppose it holds for $n = k$. We prove it for $n = k + 1$.
$A \in T_\Pi^{k+1}(\emptyset, \overline{I^+})$
$\Leftrightarrow \exists r \in ground(\Pi)$ with $head(r) = A$ and $\langle T_\Pi^k(\emptyset, \overline{I^+}), \overline{I^+} \rangle \models_{3,L} l$ for every $l \in body(r)$
$\Leftrightarrow \exists r \in ground(\Pi)$ with $head(r) = A$ and $\langle \mathcal{T}_\Pi^k(\emptyset, I^+), \overline{I^+} \rangle \models_{3,L} l$ for every $l \in body(r)$ by the induction hypothesis
$\Leftrightarrow \exists r \in ground(\Pi)$ with $head(r) = A$ and $(\mathcal{T}_\Pi^k(\emptyset, I^+), I^+) \models_L l$ for every $l \in body(r)$ by Lemma 9.13
$\Leftrightarrow A \in \mathcal{T}_\Pi^{k+1}(\emptyset, I^+)$. □

Theorem 9.18: Let Π be a normal DL logic program relative to a DL knowledge base L and I^+ a two-valued answer set of Π. Then Π has a three-valued answer set $\langle I^+, I^-\rangle$ for some $I^- \subseteq HB_\Pi$.

Proof. Assume that for any $I^- \subseteq HB_\Pi$ with $I^+ \cap I^- = \emptyset$, $\langle I^+, I^-\rangle$ is not a three-valued answer set of Π. It follows that for some $A \in I^+$, $A \notin T^\alpha_\Pi(\emptyset, I^-)$ and $L \cup T^\alpha_\Pi(\emptyset, I^-)|_\Omega \cup \neg I^- \not\models A$. Note that $\langle I^+, \overline{I^+}\rangle$ is a three-valued model of Π by Theorem 9.5. Due to Lemma 9.17, we have $T^\alpha_\Pi(\emptyset, \overline{I^+}) = \mathcal{T}^\alpha_\Pi(\emptyset, I^+)$. Since I^+ is a two-valued answer set of Π, either $A \in \mathcal{T}^\alpha_\Pi(\emptyset, I^+)$ or $L \cup \mathcal{T}^\alpha_\Pi(\emptyset, I^+)|_\Omega \cup \neg\overline{I^+}|_\Omega \models A$, a contradiction. □

Theorem 9.24: The revision operator $*^n_{\gamma^1}$ satisfies the postulates ($*^h$1b) and ($*^h$5b).

Proof. ($*^h$1b): Follows directly from Definition 9.22.
($*^h$5b): Let Π_2 be satisfiable. For any $\pi \cup \Pi_{can}(I_2) \in \Pi_1 \downarrow \Pi_2$, since $\Pi_{can}(I_2) \cup \Pi_2$ is satisfiable by Definition 9.20 and $\pi \cup \Pi_{can}(I_2) \cup \Pi_2$ is satisfiable by Definition 9.21, it holds that $\gamma^1(\Pi_1 \downarrow \Pi_2) +^h \Pi_2$ is satisfiable. □

Lemma 9.28: Let $K, K' \in \mathcal{K}_\mathcal{L}$ and γ be a selection function for K. If $\kappa \subseteq K \cap K'$, then $\kappa \subseteq \bigcap \gamma(\mathbb{K}_{K'})$.

Proof. Let $K, K' \in \mathcal{K}_\mathcal{L}$. Assume there exists $\kappa \subseteq (K \cap K') \setminus (\bigcap \gamma(\mathbb{K}_{K'}))$. Then there exists $\kappa' \in \gamma(\mathbb{K}_{K'}) : \kappa \not\subseteq \kappa'$. It follows that $QHT(\kappa' \cup \kappa) \cap SE(K') \neq \emptyset$ since $QHT(\kappa') \cap SE(K') \neq \emptyset$ by Definition 5.1 and $\kappa \subseteq K'$ implies $QHT(K') \subseteq QHT(\kappa)$. This is a contradiction because κ' is maximal by Definition 5.1. □

Theorem 9.29: Let $K \in \mathcal{K}_\mathcal{L}$. An operator $*^h_\gamma$ is a partial meet revision operator for K generated by some selection function γ for K iff $*^h_\gamma$ satisfies ($*^h$1b)–($*^h$5b).

Proof. We first show that a partial meet revision operator $*^h_\gamma$ for K generated by some selection function γ for K satisfies ($*^h$1b)–($*^h$5b).
($*^h$1b): Follows directly from Definition 9.27.
($*^h$2b): If K' is not satisfiable, then $K *^h_\gamma K' = K +^h K$. Otherwise, since $\bigcap \gamma(\mathbb{K}_{K'}) \subseteq K$ we have $\bigcap \gamma(\mathbb{K}_{K'}) +^h K' \subseteq K +^h K'$.
($*^h$3b): Let $\kappa \subseteq K$. Assume that for all κ' with $K *^h_\gamma K' \subseteq \kappa' \subset K +^h K'$ and κ' being satisfiable, it holds that $\kappa' \cup \kappa$ is satisfiable. In particular, for each $\kappa'' \in \mathbb{K}_{K'}$ with $K *^h_\gamma K' \subseteq \kappa'' \cup K'$, this implies $\kappa'' \cup K' \cup \kappa$ is satisfiable. As

each κ'' is $\subseteq_P/\subseteq_T/\subseteq$-maximal, it follows that $\kappa \subseteq \kappa''$ and thus $\kappa \subseteq \bigcap \gamma(\mathbb{K}_{K'})$. From Definition 9.27 we can then conclude $\kappa \not\subseteq K \setminus (K *_\gamma^h K')$.
($*^h$4b): For all $\kappa \subseteq K$, let $\kappa + K'$ be satisfiable iff $\kappa + K''$ is satisfiable. Then $\mathbb{K}_{K'} = \mathbb{K}_{K''}$ by Definition 9.26 and so $\bigcap \gamma(\mathbb{K}_{K'}) = \bigcap \gamma(\mathbb{K}_{K''})$ as well as $K \cap \bigcap \gamma(\mathbb{K}_{K'}) = K \cap \bigcap \gamma(\mathbb{K}_{K''})$. By Lemma 9.28 we obtain $(K \cap \bigcap \gamma(\mathbb{K}_{K'})) \cup (K \cap K') = (K \cap \bigcap \gamma(\mathbb{K}_{K''})) \cup (K \cap K'')$. This means $K \cap (\bigcap \gamma(\mathbb{K}_{K'}) \cup K') = K \cap (\bigcap \gamma(\mathbb{K}_{K''}) \cup K'')$. Thus, $K \cap (K *_\gamma^h K') = K \cap (K *_\gamma^h K'')$.
($*^h$5b): If K' is satisfiable, then for any $\kappa \in \mathbb{K}_{K'}$, $\kappa +^h K'$ is satisfiable, which implies $K *_\gamma^h K'$ is satisfiable.

We now show that any operator $\circ_\gamma^h$ for K satisfying ($*^h$1b)–($*^h$5b) is a partial meet revision operator for K generated by some selection function for K. We first find a selection function γ for K. Let γ be such that (i) if $\mathbb{K}_{K'} = \emptyset$, then $\gamma(\mathbb{K}_{K'}) = \emptyset$ and (ii) $\gamma(\mathbb{K}_{K'}) = \{\, \kappa \in \mathbb{K}_{K'} \mid K \cap (K \circ_\gamma^h K') \subseteq \kappa \,\}$ otherwise.

We begin by showing that γ is a function. If $\mathbb{K}_{K'} = \mathbb{K}_{K''}$, then $K \cap (K \circ_\gamma^h K') = K \cap (K \circ_\gamma^h K'')$ by ($*^h$4b). This means $\gamma(\mathbb{K}_{K'}) = \gamma(\mathbb{K}_{K''})$ according to our definition of γ.

We next show that γ is a selection function. Clearly, $\gamma(\mathbb{K}_{K'}) \subseteq \mathbb{K}_{K'}$ by our definition of γ. If $\mathbb{K}_{K'} \neq \emptyset$, then K' is satisfiable by Definition 5.1 and thus $K \circ_\gamma^h K'$ is satisfiable by ($*^h$5b). Since $K' \subseteq K \circ_\gamma^h K'$ by ($*^h$1b) and $K \circ_\gamma^h K' \subseteq K \cup K'$ by ($*^h$2b), it follows that $(K \cap (K \circ_\gamma^h K')) \cup K'$ is satisfiable. This means that there exists $\kappa \in \mathbb{K}_{K'}$ such that $K \cap (K \circ_\gamma^h K') \subseteq \kappa$. From our definition of γ we therefore obtain that $\gamma(\mathbb{K}_{K'}) \neq \emptyset$.

Finally, we show that $\circ_\gamma^h$ is a partial meet revision operator for K, that is, $K \circ_\gamma^h K' = K \cup K'$ if K' is not satisfiable and $K \circ_\gamma^h K' = \bigcap \gamma(\mathbb{K}_{K'}) \cup K'$ otherwise. Consider first the limiting case that K' is not satisfiable. If $\kappa \subseteq K \setminus (K \circ_\gamma^h K')$, then there exists κ' such that $K \circ_\gamma^h K' \subseteq \kappa' \subset K \cup K'$ and κ' is satisfiable but $\kappa' \cup \kappa$ is not satisfiable by ($*^h$3b). This is a contradiction since $K' \subseteq \kappa'$ by ($*^h$1b). Therefore, it holds for all $\kappa \subseteq K$ that $\kappa \subseteq K \circ_\gamma^h K'$, that is, $K \subseteq K \circ_\gamma^h K'$. Since $K' \subseteq K \circ_\gamma^h K'$ by ($*^h$1b) and $K \circ_\gamma^h K' \subseteq K \cup K'$ by ($*^h$2b), we can conclude $K \circ_\gamma^h K' = K \cup K'$.

Assume now that K' is satisfiable. Let $\kappa \subseteq K \setminus (K \circ_\gamma^h K')$. If $\mathbb{K}_{K'} = \emptyset$, then it follows from ($*^h$1b) and ($*^h$3b) that $K \circ_\gamma^h K' = K'$. Since $\gamma(\mathbb{K}_{K'}) = \emptyset$ by our definition of γ, we thus have $K \circ_\gamma^h K' = K' = \bigcap \gamma(\mathbb{K}_{K'}) \cup K'$. If $\mathbb{K}_{K'} \neq \emptyset$, then it follows directly from our definition of γ that $K \cap (K \circ_\gamma^h K') \subseteq \bigcap \gamma(\mathbb{K}_{K'})$. From ($*^h$1b) and ($*^h$2b) we then obtain $K \circ_\gamma^h K' \subseteq \bigcap \gamma(\mathbb{K}_{K'}) \cup K'$. To show the converse inclusion, first assume the case that $K \cup K'$ is satisfiable. This

implies that for any $\kappa' \subseteq K \cup K'$ it holds that κ' is satisfiable. Applying ($*^h$3b), we obtain $K \setminus (K \circ^h_\gamma K') = \emptyset$ and thus $K \subseteq K \circ^h_\gamma K'$. From ($*^h$1b) and ($*^h$2b) it follows that $K \circ^h_\gamma K' = K \cup K'$. Moreover, due to the assumption that $K \cup K'$ is satisfiable and Definition 5.1, we have $\mathbb{K}_{K'} = \{K\}$. By our definition of γ, we obtain $\gamma(\mathbb{K}_{K'}) = \{K\}$ and thus $\bigcap \gamma(\mathbb{K}_{K'}) = K$ and can conclude $K \circ^h_\gamma K' = \bigcap \gamma(\mathbb{K}_{K'}) \cup K'$. Lastly, assume the case that $K \cup K'$ is not satisfiable. We will show that $\kappa \notin K \circ^h_\gamma K'$ implies $\kappa \notin \bigcap \gamma(\mathbb{K}_{K'}) \cup K'$. If $\kappa \notin K$, then $\kappa \notin (K \circ^h_\gamma K') \setminus K'$ by ($*^h$1b) and ($*^h$2b) and $\kappa \notin \bigcap \gamma(\mathbb{K}_{K'})$ by Definition 5.1. Since $\kappa \notin K \circ^h_\gamma K'$ implies $\kappa \notin K'$ by ($*^h$1b), it follows that $\kappa \notin ((K \circ^h_\gamma K') \setminus K') \cup K') = K \circ^h_\gamma K'$ and $\kappa \subseteq \bigcap \gamma(\mathbb{K}_{K'}) \cup K'$. Now assume $\kappa \subseteq K \setminus (K \circ^h_\gamma K')$. According to ($*^h$3b), then there exists κ' such that $K \circ^h_\gamma K' \subseteq \kappa' \subset K \cup K'$ and κ' is satisfiable but $\kappa' \cup \kappa$ is not satisfiable. This means that there exists $\kappa'' \in \mathbb{K}_{K'}$ such that $K \cap \kappa' \subseteq \kappa''$ and $\kappa \notin \kappa''$. Since $K \cap (K \circ^h_\gamma K') \subseteq K \cap \kappa' \subseteq \kappa''$, we obtain from our definition of γ that $\kappa'' \in \gamma(\mathbb{K}_{K'})$. We can thus conclude from $\kappa \notin \kappa''$ that $\kappa \notin \bigcap \gamma(\mathbb{K}_{K'})$. □

Theorem 9.31: Let $K \in \mathcal{K}_\mathcal{L}$. An operator $\dot{-}^h_\gamma$ is a partial meet contraction operator for K generated by some selection function γ for K iff $\dot{-}^h_\gamma$ satisfies ($\dot{-}^h$1b)–($\dot{-}^h$4b).

Proof. ($\dot{-}^h$1b): Follows directly from Definition 9.30.

($\dot{-}^h$2b): Let $\not\models K'$. For any $\kappa \in \mathbb{K}^-_{K'}$, $\kappa \not\models K'$, which implies $K \dot{-}^h_\gamma K' \not\models K'$.

($\dot{-}^h$3b): Let $\kappa \subseteq K$. Assume that for all κ' with $K \dot{-}^h_\gamma K' \subseteq \kappa' \subseteq K$ and $\kappa' \not\models K'$, it holds that $\kappa' \cup \kappa \not\models K'$. In particular, for each $\kappa' \in \mathbb{K}^-_{K'}$ with $K \dot{-}^h_\gamma K' \subseteq \kappa'$, this implies $\kappa' \cup \kappa \not\models K'$. As each κ' is $\subseteq_P/\subseteq_T/\subseteq$-maximal, it follows that $\kappa \subseteq \kappa'$ and thus $\kappa \subseteq K \dot{-}^h_\gamma K'$.

($\dot{-}^h$4b): For all $\kappa' \subseteq K$, let $\kappa' \not\models K'$ iff $\kappa' \not\models K''$. Then $\mathbb{K}^-_{K'} = \mathbb{K}^-_{K''}$ by definition of $\mathbb{K}^-_{K'}$ and so $\gamma(\mathbb{K}^-_{K'}) = \gamma(\mathbb{K}^-_{K''})$ as well as $\bigcap \gamma(\mathbb{K}^-_{K'}) = \bigcap \gamma(\mathbb{K}^-_{K''})$. Thus, $K \dot{-}_\gamma K' = K \dot{-}_\gamma K''$ by Definition 9.30.

We now show that any operator $\circ^h_\gamma$ for K satisfying ($\dot{-}^h$1b)–($\dot{-}^h$4b) is a partial meet contraction operator for K generated by some selection function for K. We first find a selection function γ for K. Let γ be such that (i) if $\mathbb{K}^-_{K'} = \emptyset$, then $\gamma(\mathbb{K}^-_{K'}) = \emptyset$ and (ii) $\gamma(\mathbb{K}^-_{K'}) = \{\, \kappa \in \mathbb{K}^-_{K'} \mid K \circ^h_\gamma K' \subseteq \kappa \,\}$ otherwise.

We begin by showing that γ is a function. If $\mathbb{K}^-_{K'} = \mathbb{K}^-_{K''}$, then $K \circ^h_\gamma K' = K \circ^h_\gamma K''$ by ($\dot{-}^h$4b). This means $\gamma(\mathbb{K}^-_{K'}) = \gamma(\mathbb{K}^-_{K''})$ according to our definition of γ.

We next show that γ is a selection function. Clearly, $\gamma(\mathbb{K}^-_{K'}) \subseteq \mathbb{K}^-_{K'}$ by our definition of γ. If $\mathbb{K}^-_{K'} \neq \emptyset$, then $\not\models K'$ by the definition of $\mathbb{K}^-_{K'}$ and thus $K \circ^h_\gamma K' \not\models K'$ by ($\dot{-}^h$2b). It follows from $K \circ^h_\gamma K' \subseteq K$ due to ($\dot{-}^h$1b) that there exists $\kappa \in \mathbb{K}^-_{K'}$ such that $K \circ^h_\gamma K' \subseteq \kappa$. From our definition of γ we therefore obtain that $\gamma(\mathbb{K}^-_{K'}) \neq \emptyset$.

Finally, we show that $\circ^h_\gamma$ is a partial meet contraction operator for K, that is, $K \circ^h_\gamma K' = K$ if $\models K'$ and $K \circ^h_\gamma K' = \bigcap \gamma(\mathbb{K}^-_{K'})$ otherwise. Consider first the limiting case that $\models K'$. If $\kappa \subseteq K \setminus (K \circ^h_\gamma K')$, then there exists κ' such that $K \circ^h_\gamma K' \subseteq \kappa' \subset K$ and $\kappa' \not\models K'$ but $\kappa' \cup \kappa \models K'$ by ($\dot{-}^h$3b). This is a contradiction since $\models K'$. Therefore, it holds for all $\kappa \subseteq K$ that $\kappa \subseteq K \circ^h_\gamma K'$, that is, $K \subseteq K \circ^h_\gamma K'$. Since $K \circ^h_\gamma K' \subseteq K$ by ($\dot{-}^h$1b), we can conclude $K \circ^h_\gamma K' = K$.

Assume now that $\not\models K'$. Let $\kappa \subseteq K \setminus (K \circ^h_\gamma K')$. If $\mathbb{K}^-_{K'} = \emptyset$, then it follows from ($\dot{-}^h$2b) and ($\dot{-}^h$3b) that $K \circ^h_\gamma K' = \emptyset$. Since $\gamma(\mathbb{K}^-_{K'}) = \emptyset$ by our definition of γ, we thus have $K \circ^h_\gamma K' = \bigcap \gamma(\mathbb{K}^-_{K'})$. If $\mathbb{K}^-_{K'} \neq \emptyset$, then it follows directly from our definition of γ that $K \circ^h_\gamma K' \subseteq \bigcap \gamma(\mathbb{K}^-_{K'})$. To show the converse inclusion, first assume the case that $K \not\models K'$. This implies that for any $\kappa' \subseteq K$ it holds that $\kappa' \not\models K'$. Applying ($\dot{-}^h$3b), we obtain $K \setminus (K \circ^h_\gamma K') = \emptyset$ and thus $K \subseteq K \circ^h_\gamma K'$. From ($\dot{-}^h$1b) it follows that $K \circ^h_\gamma K' = K$. Moreover, due to the assumption that $K \not\models K'$ and the definition of $\mathbb{K}^-_{K'}$, we have $\mathbb{K}^-_{K'} = \{K\}$. By our definition of γ, we obtain $\gamma(\mathbb{K}^-_{K'}) = \{K\}$ and thus $\bigcap \gamma(\mathbb{K}^-_{K'}) = K$ and can conclude $K \circ^h_\gamma K' = \bigcap \gamma(\mathbb{K}^-_{K'})$. Lastly, assume the case that $K \models K'$. We will show that $\kappa \not\subseteq K \circ^h_\gamma K'$ implies $\kappa \not\subseteq \bigcap \gamma(\mathbb{K}^-_{K'})$. If $\kappa \not\subseteq K$, then $\kappa \not\subseteq K \circ^h_\gamma K'$ by ($\dot{-}^h$1b) and $\kappa \not\subseteq \bigcap \gamma(\mathbb{K}^-_{K'})$ by the definition of $\mathbb{K}^-_{K'}$. Now assume $\kappa \subseteq K \setminus (K \circ^h_\gamma K')$. According to ($\dot{-}^h$3b), then there exists κ' such that $K \circ^h_\gamma K' \subseteq \kappa' \subset K$ and $\kappa' \not\models K'$ but $\kappa' \cup \kappa \models K'$. This means that there exists $\kappa'' \in \mathbb{K}^-_{K'}$ such that $\kappa' \subseteq \kappa''$ and $\kappa \not\subseteq \kappa''$. Since $K \circ^h_\gamma K' \subseteq \kappa' \subseteq \kappa''$, we obtain from our definition of γ that $\kappa'' \in \gamma(\mathbb{K}^-_{K'})$. We can thus conclude from $\kappa \not\subseteq \kappa''$ that $\kappa \not\subseteq \bigcap \gamma(\mathbb{K}^-_{K'})$. □