KU LEUVEN

ARENBERG DOCTORAL SCHOOL
Faculty of Engineering Science

Groundedness in Logics With a Fixpoint Semantics

Bart Bogaerts

Supervisor:
Prof. Dr. M. Denecker
Prof. Dr. J. Vennekens, co-supervisor
Prof. Dr. J. Van den Bussche, co-supervisor

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor in Engineering

June 2015

Groundedness in Logics With a Fixpoint Semantics

Bart BOGAERTS

Examination committee:
Prof. Dr. ir. J. Vandewalle, chair
Prof. Dr. M. Denecker, supervisor
Prof. Dr. J. Vennekens, co-supervisor
Prof. Dr. J. Van den Bussche, co-supervisor
Prof. Dr. ir. M. Bruynooghe
Prof. Dr. ir. F. Piessens
Prof. Dr. Gerhard Brewka
(University of Leipzig)
Prof. Dr. Thomas Eiter
(Vienna University of Technology)

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor in Engineering

June 2015

Abstract

In the field of *knowledge representation and reasoning*, many different logics are developed. Often, these logics exhibit striking similarities, either because they emerged from related ideas, or because they use similar underlying fundamental principles. *Approximation fixpoint theory (AFT)* is an abstract algebraical unifying framework that aims at exposing these principles by formalising them in lattice theory. It has been successfully applied to unify all common semantics of logic programs, autoepistemic logic, default logic, and more recently Dung's argumentation frameworks and abstract dialectical frameworks.

In this dissertation, we extend approximation fixpoint theory to expose more underlying principles common to the aforementioned logics. In these domains, researchers have made use of a similar intuition: that facts (or models) can be derived *from the ground up*. They typically phrase this intuition by saying, e.g., that the facts should be *grounded*, or that they should not be *unfounded*, or that they should be supported by *cycle-free* arguments. In different domains, semantics that allow *ungrounded* models have received a lot of criticism. In logic programming for example, this was the case for Clark's completion semantics, which was later improved by perfect model semantics, stable semantics and well-founded semantics. In autoepistemic logic, a similar evolution happened: Moore's expansion semantics turned out to allow self-supporting models; this resulted in the development of many different semantics in attempts to get rid of this erroneous behaviour.

In the first part of this dissertation, we formalise groundedness in approximation fixpoint theory. We study how groundedness relates to other concepts and fixpoints studied in AFT. We apply our abstract theory to the aforementioned domains: we show that our notion of groundedness indeed captures the intuitions that existed in these domains and study complexity of reasoning with grounded models. We study which existing semantics are grounded and which are not. For example, for logic programming, we find that Clark's completion semantics (indeed) is not grounded, while stable and well-founded semantics are grounded.

We show that the well-founded model is not just any grounded model: it is the least precise partial grounded model.

In the second part of this thesis we define a class of autoepistemic theories for which it is informally clear how to construct the intended model. Unfortunately, despite previous claims that the well-founded semantics captures the meaning of autoepistemic theories very well (e.g., because of its constructive nature), the well-founded semantics fails to identify this model. In order to overcome this limitation, we propose, algebraically, a new constructive semantics based on the notion of groundedness. Our new construction refines the well-founded model construction and succeeds in identifying the intended model for the class of motivating examples. Furthermore, we show that for this class of examples, our novel construction constructs the unique grounded fixpoint.

Summarised, in this dissertation, we continue the work on approximation fixpoint theory by identifying novel concepts occurring in all of the application domains and by refining existing semantics to better capture the intended meaning of a class of theories.

Beknopte samenvatting

In het onderzoeksdomein *kennisrepresentatie en redeneren* worden formele talen, ook wel *logicas* genoemd, ontwikkeld met als doel kennis op een natuurlijke manier neer te kunnen schrijven en deze kennis te gebruiken om automatisch te redeneren. Verschillende logicas vertonen vaak sterke gelijkenissen, doordat ze uit gelijkaardige ideeën ontstaan, of doordat ze op dezelfde fundamentele principes gebaseerd zijn. *Approximation fixpoint theory (AFT)* is een abstracte algebraïsche theorie waarin dergelijke onderliggende principes expliciet worden gemaakt, door ze op een hoog niveau, los van de logicas zelf, te formaliseren. AFT is reeds succesvol toegepast om de verschillende semantieken van *logisch programmeren*, *autoepistemische logica*, *default logica* en *abstracte argumentatietheorie* te unificeren.

In dit proefschrift breiden we AFT uit om meer onderliggende principes bloot te leggen. In de verschillende hogervermelde domeinen hebben onderzoekers uitspraken gedaan over een gelijkaardige intuïtie. Ze verwezen vaak naar het feit dat modellen *gegrond* (*grounded*) moeten zijn, of dat modellen *van de grond af aan* opgebouwd kunnen worden of *geen vicieuze cirkels* mogen bevatten. In de verschillende onderzoeksdomeinen werden modellen die deze—informele—eigenschap niet hebben steeds als slecht beschouwd. Bijvoorbeeld in logisch programmeren was er veel kritiek op Clark's zogenaamde *completion semantiek* om precies die reden. Dit heeft geleid tot de ontwikkeling van verschillende verbeterde semantieken die enkel *gegronde* modellen toelaten. In autoepistemische logica vond een gelijkaardige ontwikkeling plaats: Moore's oorspronkelijke semantiek liet ongegronde modellen toe en verschillende onderzoekers ontwikkelden betere semantieken die dit gedrag niet vertonen.

In het eerste deel van deze tekst formaliseren we het begrip *gegrondheid* in AFT. We geven een abstract algebraïsche definitie van deze notie en bestuderen hoe gegrondheid zich verhoudt tot andere concepten die in AFT bestudeerd worden. Nadien passen we onze theorie toe op de verschillende logicas. We tonen aan dat onze abstracte definitie inderdaad bestaande intuïties kan vatten. We

bestuderen in elk van de verschillende domeinen welke bestaande semantieken al dan niet gegrond zijn. Bovendien geven we een alternatieve definitie van de well-founded semantiek gebaseerd op gegrondheid.

In het tweede deel van deze thesis gebruiken we gegrondheid om een ander probleem op te lossen. We tonen aan dat er in autoepistemische logica een klasse van theorieën is, waarvoor de well-founded semantiek ontoereikend is, ondanks sterke eerdere claims dat de well-founded semantiek de betekenis van autoepistemische theorieën zeer goed vat. We definiëren dan, opnieuw algebraïsch, een nieuwe constructieve semantiek gebaseerd op gegrondheid. Wanneer we deze algebraïsche constructie toepassen op autoepistemische logica, krijgen we een semantiek die wel het gewenste resultaat geeft op de voorgenoemde klasse van theorieën.

Kort samengevat zetten we in dit proefschrift het werk rond AFT verder door nieuwe concepten te identificeren die in verschillende logische domeinen voorkomen en door een bestaande algebraïsche constructie te verfijnen zodat ze het intuïtief verwachte resultaat produceert op een grotere klasse van theorieën.

Preface

Today is the end of a 4 year lasting chapter of my life. I've had a really great time, filled with challenging questions, opportunities to meat interesting people and a lot of liberty to choose what to work on. Still, it feels good to close this chapter and start a new one. There are many people who contributed to making these four years enjoyable; for that, I'm grateful.

First of all, I want to thank my **supervisors**. I started this PhD with just one supervisor, Marc. Over the past four years, he has been a non-stop source of (mostly great) research ideas; if I had to work out all of his suggestions before graduating, my PhD would take the rest of my life. All of the work in this dissertation is somehow based on ideas he came up with. Marc, thank you for turning me into a researcher, for all the constructive feedback on my texts[1], and the countless interesting discussions and coffee breaks! When my research evolved, also Joost and Jan became my supervisors. Together with Marc, they formed a great complimentary team. While Marc was dreaming about drastically changing the entire field of computational logic, Joost often managed to stay grounded[2] and take a nuanced position. Joost also has the ability to turn entire paragraphs into one sentence (with a greater information content). While Marc spent hours to perfect the abstract of a paper, Jan was able to uncover every tiny mathematical mistake or abuse of notation after the first reading of a text. Jan is incredibly fast, and has given me great advice on how to "sell" my papers. Marc, Joost, Jan, it goes without saying that I could not have written this dissertation without your help. Thank you!

I want to thank all my **colleagues** for the great scientific collaborations, the endless[3] discussions at our whiteboard and the legendary board game sessions. But most importantly, thanks for forming an extremely cohesive research group:

[1] Even though the perfect moment to send this feedback seems to be less than 24 hours before the submission deadline.

[2] A property that was of great importance for this dissertation!

[3] Literally, endless.

whenever I needed someone to get feedback on some new ideas, to proofread a draft, to explain some new C++11 features, or to answer some nagging question, you guys were there!

I am grateful to the members of my **jury**, who found time in their busy schedule to read (earlier versions of) this text and suggest countless improvements. Thanks for the suggestions, for challenging me, and for coming to Leuven for my defence.

In Dutch there's a saying "In nood leert men zijn vrienden kennen[4]". The last few months were such a period "in need" during which my wife and I spent more time in the hospital than at home. We tested the saying and are grateful to all our **friends** and **family**[5]. Thanks for helping us get through this tough period. Thanks for being there, no matter what, and asking if there's anything you could do...

Finally (last but definitely not least), I want to dedicate this text to two special people. **Evelyn**, my sweetheart, thanks for making our apartment a warm home to return to every day. Thanks for being there, thanks for everything! **Felientje**, our cute little girl (and primary source of sleep deprivation), keep on fighting. We love you!

[4]This translates roughly to "a friend in need, a friend indeed".

[5]Special thanks goes out to my parents and parents-in-law who basically took over our entire household.

Acronyms

ADF abstract dialectical framework.

AEL autoepistemic logic.

AF argumentation framework.

AFT approximation fixpoint theory.

AI artificial intelligence.

ASP answer set programming.

DL default logic.

DNF disjunctive normal form.

FO first-order logic.

KBS knowledge base system.

KRR knowledge representation and reasoning.

LP logic programming.

LTC linear time calculus.

OEL ordered epistemic logic.

List of Symbols

A	An approximator of O, page 12
$A_{\mathcal{T}}$	The standard approximator of $D_{\mathcal{T}}$, page 48
$At_O(\varphi)$	The atoms occurring objectively in φ, page 67
$At(\varphi)$	The atoms occurring in φ, page 67
$body(r)$	The body of rule r, page 14
$\bot$	The least element of a lattice, page 11
$C(X)$	An abstract constraint, page 33
C_s^{in}	The set conditions for s to be accepted in an ADF, page 39
$\Delta_{\mathcal{T}}$	The semantic operator of an autoepistemic theory, page 45
$\equiv$	An equivalence relation, page 83
$\mathbf{f}$	The truth value "false", page 14
F_Θ	The characteristic function of Θ, page 38
$gf(O)$	The set of grounded fixpoints of O, page 81
$\bigwedge S$	The greatest lower bound of S, page 11
$\mathrm{glb}(S)$	The greatest lower bound of S, page 11
G_Ξ	The semantic operator of Ξ, page 40
$head(r)$	The head of rule r, page 14
$\mathcal{I}$	A four-valued interpretation, page 15
I, J	Interpretations, page 14

K	The autoepistemic operator, page 7
$kks(O)$	The Kripke-Kleene set of O, page 74
Kon	Konolige's mapping from DL to AEL, page 50
$\mathcal{L}$	The language of propositional logic, page 43
L	A lattice, page 11
L^2	The bilattice of L, page 12
L^c	The consistent elements of L^2, page 12
$\leq_f$	The "finer" relation between equivalence relations, page 86
$\leq_k$	The knowledge order, page 44
$\leq_p$	The precision ordering on L^2, page 12
$\text{lfp}(O)$	The least fixpoint of O, page 12
$\mathcal{L}_K$	The language of modal propositional logic, page 43
$\bigvee S$	The least upper bound of S, page 11
$\text{lub}(S)$	The least upper bound of S, page 11
$\models$	The satisfaction relation, page 44
$\models_{gf}$	The grounded fixpoint satisfaction relation, page 94
$\models_{wfs}$	The well-founded set satisfaction relation, page 94
O	A lattice operator, page 11
$O_{\equiv}$	The induced operator on the quotient lattice, page 88
$\mathcal{P}$	A logic program, page 14
$par(s)$	The parents of S in an ADF, page 39
$p_{\equiv}$	The quotient mapping, page 83
φ	A formula, page 14
$\oplus$	The extension of possible world structures, page 68
$pm(\mathcal{T})$	The perfect model of $\mathcal{T}$, page 71
$\Psi_{\mathcal{P}}$	Fitting's partial immediate consequence operator of $\mathcal{P}$, page 15

Σ	An alphabet, page 14
$\mathbf{t}$	The truth value "true", page 14
$\mathcal{T}$	A theory, page 43
$T_{\mathcal{P}}$	The immediate consequence operator of $\mathcal{P}$, page 14
Θ	An abstract argumentation framework, page 37
$Th_{obj}(Q)$	The objective theory of Q, page 44
$\top$	The greatest element of a lattice, page 11
$\overline{U}$	The complement of U, page 29
U_O	The ultimate approximator of O, page 14
$U_{\mathcal{T}}$	The ultimate approximator of $D_{\mathcal{T}}$, page 68
U_Θ	The unattacked mapping of Θ, page 38
$wfs(O)$	The well-founded set of O, page 79
$\mathcal{W}_\Sigma$	The set of all possible world structures over Σ, page 43
$x_\equiv$	The equivalence class of x, page 83
Ξ	An abstract dialectical framework, page 39

Contents

Chapter 1

Introduction

1.1 Knowledge Representation and Reasoning

This thesis is situated in knowledge representation and reasoning (KRR) (Baral, 2003). KRR is a subfield of artificial intelligence (AI) concerned with defining languages to represent knowledge and devising methodologies and tools to reason with this knowledge. Formal studies of knowledge representation and reasoning with knowledge date back to ancient times. As an example, consider Aristotle's famous syllogism

> "All men are mortal.
>
> Socrates is a man.
>
> Therefore, Socrates is mortal."

The argument presented above is inarguably valid. The three above sentences are now commonly represented in first-order logic (FO):

$$\forall x : Man(x) \Rightarrow Mortal(x).$$

$$Man(Socrates).$$

$$Mortal(Socrates).$$

While the Greeks relied on philosophers to make (formal) arguments like the one above, thanks to the progress in computer science, a simple personal computer can now automatically infer that the last sentence is indeed a consequence of the first two.

KRR is concerned with a broad variety of tasks; FO is only one of the many languages to represent knowledge in and deduction, which is the kind of reasoning used above, is only one of many reasoning methods that are studied. The following are some major research topics in KRR. The list is non-exhaustive, and in no particular order.

Defining knowledge representation languages While statements of the form "All men are mortal." are easily expressed in FO, other forms of knowledge cannot or not easily be expressed in this language. For this kind of knowledge, we need to search for better, or extended formal languages.

Consider for example the concept of *transitive closure*. The transitive closure TC_E of a graph E is defined as the set of pairs of nodes (x, y) such that there is a path from x to y in E. It is well-known that this relation cannot be expressed in general in first-order logic. In order to overcome this limitation, several authors have studied the concept of *inductive definitions* (Post, 1943; Spector, 1961; Kreisel, 1963; Feferman, 1970; Martin-Löf, 1971; Moschovakis, 1974a,b; Aczel, 1977; Buchholz et al., 1981; Hallnäs, 1991). Inductive definitions, and related fixpoint constructs have been integrated with first-order logic in many languages such as μ-calculus (Kozen, 1983; Streett and Emerson, 1989), database query languages (Afanasiev et al., 2008; Bidoit and Ykhlef, 1998), description logics (Calvanese et al., 1999) and in the logic FO(ID) (Denecker and Ternovska, 2008). In FO(ID), the transitive closure of a road network contains all tuples of cities x and y such that it is possible to drive from x to y; using inductive definitions, this relation is expressed by the following rule set

$$\left\{ \begin{array}{l} \forall x, y : Reach(x, y) \leftarrow Road(x, y). \\ \forall x, y : Reach(x, y) \leftarrow \exists z : Road(x, z) \wedge Reach(z, y). \end{array} \right\}$$

This rule set is read intuitively as follows: "City y is reachable from city x if there is a road from x to y, or if there is a road from x to another city (z) from which y is reachable. In all other cases, y is not reachable from x". Extensions of first-order logic, with constructs such as for example a transitive closure form the basis of many database languages (Abiteboul et al., 1995).

All knowledge we saw this far was *objective*: it consists of statements about the real world. In some contexts, such as for example games with incomplete information, it is useful to reason about subjective information, e.g., about what an opponent knows and what he does not know. This kind of information also cannot be expressed in FO. In order to express such knowledge, new logics with special language constructs to refer to knowledge are developed. For example the logical formula

$$(Play(O, Hearts) \wedge \neg FollowSuit(I)) \Rightarrow K_O \neg Have(I, Hearts)$$

is read "If my opponent plays hearts, and I do not follow suit, then my opponent knows that I do not have hearts". The operator K_O is an *epistemic* operator: it refers to the knowledge of the agent O, my opponent (Fagin et al., 1995).

Studying the different forms of knowledge humans possess and defining formalisms to express this knowledge unambiguously is an important task in KRR. Popular knowledge representation languages include description logics (Baader et al., 2003), answer set programming (ASP) (Marek and Truszczyński, 1999; Lifschitz, 1999; Niemelä, 1999; Eiter et al., 2009; Gebser et al., 2012), argumentation frameworks (Dung, 1995), and extensions of first-order logic (Denecker, 2012). Some languages focus on the ability to describe a large variation of problem domains, while other languages, such as for example situation calculus (Levesque et al., 1998; Pirri and Reiter, 1999; Reiter, 2001) and event calculus (Shanahan, 1997), focus on expressing knowledge in a specific setting; the two aforementioned languages focus on describing temporal domains.

Defining Inference Methods Concluding that Socrates is mortal, given the fact that he is a man and all men are mortal, is a form of inference, called *deduction*. Several other forms of inference exist.

One such inference method is *model checking*: verifying whether an interpretation satisfies a logical theory. For example, given a logical theory that describes what are valid course schedules (no two courses in the same room at the same time, all courses take place in a room with sufficient capacity, et cetera), decide whether or not some given hand-crafted schedule satisfies all constraints in the theory. Another inference method is *model expansion*, or *answer set generation* (Mitchell and Ternovska, 2005; Ternovska and Mitchell, 2009; Gebser et al., 2012): given a logical theory, and partial information about the real world, find models of this theory. E.g., with the same scheduling theory as used above, the model expansion inference consists of searching a valid schedule given some partial input, for example the set of courses, students, teachers, rooms, et cetera. Other inference methods include querying (Vardi, 1986), propagation (Wittocx et al., 2013), abduction (Peirce and Buchler, 1955) and progression (Lin and Reiter, 1997; Bogaerts et al., 2014a).

Depending on the problem at hand, different existing inference methods can be used or new inference methods might need to be developed. Some inference methods are designed to be generally applicable, while others focus on a particular type of application domain, such as for example temporal (Schnoebelen, 2003; Eén and Sörensson, 2003) or spatial (Randell et al., 1992) reasoning.

Implementing Inference Engines In order to be of practical use, the logics and inference methods should not just be defined in some mathematical book; we need automated systems that interpret the defined logics and that are able to perform inference on them. Inference engines often perform one type of inference for a specific language. For example, automated theorem provers such as SPASS (Weidenbach et al., 2009), Vampire (Riazanov and Voronkov, 2002) and many more (Sutcliffe, 2013) perform deduction[1] for (extensions of) FO, ASP-solvers (Alviano et al., 2013; Gebser et al., 2012) perform answer set generation, ... Sometimes, it is also useful to have a system that can perform many forms of inference on *the same* language. This way, we can achieve that knowledge is represented once, and can be reused to solve different types of problems in the same problem domain. This is the core idea underlying the knowledge base system (KBS) paradigm (Denecker and Vennekens, 2008), which is implemented in the IDP system (De Cat et al., 2014a).

Studying Complexity, Expressivity and Succinctness We already mentioned that certain forms of knowledge can be *inexpressible* in certain logics. Studying exactly what can and cannot be expressed in a certain logic can yield new insights in this logic. A related research question is how *succinct* knowledge can be expressed in one language.

Depending on the logic at hand, some inference methods might be simple to execute, while others might be hard, or even impossible to perform in general. In *computational complexity theory* (Papadimitriou, 1994), inference methods are classified according to their inherent difficulty. *Knowledge compilation* (Cadoli and Donini, 1997; Darwiche and Marquis, 2002) studies the relation between the expressive power of knowledge representation languages and their support for efficient inference by identifying languages that support certain queries and transformations efficiently. It studies the relative succinctness of these languages, and is concerned with building *compilers* that can transform knowledge bases into a desired target language.

Unifying Logics In different research groups, different logics are developed. Often, they contain similar ideas that are worked out slightly differently, or approached from another angle. In order to keep an overview, it is important to know how the different languages relate: are they different dialects but based on the same principles or are they fundamentally different? One possible way to study these relationships, is by defining transformations between logics, i.e., defining a semantic embedding of one logic in another. Another way is to devise

[1] Given the undecidability of deduction for first-order logic, these provers are often incomplete, or only cover fragments of the language.

unifying frameworks that capture the semantics of a large family of logics at once. Several such frameworks exist (Denecker and De Schreye, 1993; Bonatti, 1995; Denecker et al., 2000; Thielscher, 2011).

Applications Knowledge is everywhere: every aspect of human life is flooded with knowledge. For example, traffic regulations are knowledge, whether or not there are tomatoes left in the fridge is knowledge, the evolution of the stock markets over time is knowledge, the rules of the Sudoku game are knowledge, ... As such, it is to be expected that KRR has many applications in different research domains. Indeed, we find applications of KRR in robotics (Andres et al., 2013), security (Barker et al., 2014), privacy (Decroix et al., 2014), linguistics (Andrews et al., 2012), pharmacology (Prokosch et al., 1991), machine learning and data mining (Bruynooghe et al., 2015) and many more domains (van Harmelen et al., 2007). Applications of KRR help to valorise the research in this field.

1.2 Contributions to Knowledge Representation and Reasoning

My research as a PhD student has been fairly broad. Over the last four years, I have been interested in all of the above research topics. This section briefly summarises the various topics my coauthors and I worked on, and discusses our main contributions.

When performing inference, symmetries of the problem domain can be taken into account to increase efficiency. For example in a course scheduling application, two students who follow exactly the same courses are interchangeable. A valid schedule for one of these students is also a valid schedule for the other. Incorporating this kind of information in the inference engine can result in significant speedups. In order to achieve these speedups, two tasks need to be tackled, namely first *detecting* that some symmetries are present, and secondly *exploiting* the detected symmetries. We studied symmetry for SAT solvers: model expansion engines for propositional logic. We devised a dynamic symmetry breaking algorithm, called SP(SAT) (Devriendt et al., 2012), which outperforms the state-of-the art in symmetry breaking for SAT. We also devised a symmetry detection algorithm, called BreakID (Devriendt et al., 2014) and used the latter to win the hard-combinatorial track of the 2013 SAT competition (Balint et al., 2013).

We implemented the IDP knowledge base system (De Cat et al., 2014a) and its underlying solver MiniSAT(ID) (De Cat et al., 2013, 2014b). We used IDP to model and solve some machine learning and data mining applications (Bruynooghe et al., 2015). Furthermore, we bootstrapped IDP: we implemented parts of IDP using the inferences methods IDP implements itself (of course, in the process, avoiding possible loops) (Bogaerts et al., 2014b).

We defined the causal logic FO(C) (Bogaerts et al., 2014c). This logic focuses on expressing complex cause-effect relations, such as dynamic non-deterministic choice (one object among a yet to be determined group of objects is randomly selected) and object creation (certain actions can have the creation of an object as effect). We studied complexity of several inference methods in this logic (Bogaerts et al., 2014e) and compared FO(C) to other languages (Bogaerts et al., 2014d) such as FO, disjunctive ASP, and extensions of Datalog (Abiteboul and Vianu, 1991).

We defined the linear time calculus (LTC), a logic to describe temporal systems; we defined and implemented several inference methods specifically suited for these temporal domains (Bogaerts et al., 2014a). This study was, among other things, a validation of the KBS paradigm: we demonstrated the potential of the KBS approach in a temporal context.

We extended approximation fixpoint theory (AFT), an algebraical unifying theory that captures the semantics of several non-monotonic logics (cfr. Section 1.3). We extended it with *grounded fixpoints* (Bogaerts et al., 2015a). We applied this abstract theory to different logics, and used groundedness to define an improved constructive semantics for these logics. In this dissertation, we focus on the research revolving around the notion of groundedness. First, we give some more background on non-monotonic reasoning, and approximation fixpoint theory.

1.3 Non-Monotonic Reasoning

First-order logic is a monotonic formalism: adding more information to a theory always allows us to derive more information. For example, adding the information

> "Plato is a man."

to the syllogism at the start of this chapter also allows us to derive that Plato is mortal. All previous derivations remain valid. For instance, Socrates remains mortal.

Other formalisms do not have this property. Adding new information might invalidate previous derivations. In this dissertation, we focus on default logic (DL), autoepistemic logic (AEL), logic programming (LP) and abstract dialectical frameworks (ADFs). We briefly discuss the first two of these.

A *default* is a statement of the form "most P's are Q's". For example

> "Most birds can fly."

Reiter (1980) was the first to define a formal language to describe defaults. In his seminal paper, he assumes that such a statement means

> "If x is a bird, then in the absence of any evidence to the contrary, infer that x can fly."

The above sentence combined with (only) the information "Tweety is a bird" allows us to conclude that Tweety can fly. If more information is discovered, for example the fact that Tweety actually is a penguin, the above derivation is no longer valid. Thus, in the case of default logic, adding more information does not always lead to more derived information. This kind of derivation is sometimes called *defeasible inference*.

Autoepistemic logic (AEL) is a non-monotonic logic for modelling the beliefs or knowledge of a rational agent with perfect introspection capabilities (Moore, 1985). In AEL, the non-monotonicity is more explicit. In this logic, statements of the form $K\varphi$, intuitively read as "I know φ", can occur arbitrarily in logical theories. For example, a statement $\neg KP \Rightarrow Q$ is read "if I do not know P, then Q holds". In a situation where P is not known, the above sentence can be used to derive Q. It is clear that adding more knowledge may increase the truth of KP and hence may invalidate the derivation of Q.

There is a strong similarity between default and autoepistemic logic. Reiter's interpretation of the default "Most birds can fly" can be translated into autoepistemic logic as follows:

$$KBird(Tweety) \wedge \neg K \neg Fly(Tweety) \Rightarrow Fly(Tweety),$$

or "if I know that Tweety is a bird, and I do not know that it cannot fly, conclude that Tweety can fly". Konolige (1988) noted this similarity and used it to define a mapping from DL to AEL. However, contrary to the expectations, this mapping does not preserve semantics. For a long time, researchers believed that AEL and DL are really two different logics. In fact, Gottlob (1995) showed that no modular translations exist from DL to AEL (but non-modular transformations do exist). The exact relationship between DL and AEL was only discovered with the introduction of approximation fixpoint theory (Denecker et al., 2000).

1.4 Approximation Fixpoint Theory

Motivated by structural analogies in the semantics of several non-monotonic logics, Denecker, Marek and Truszczyński (from now on abbreviated as DMT) (2000) developed an algebraical fixpoint theory that defines different types of fixpoints for a so-called approximating bilattice operator. They called these fixpoints supported, Kripke-Kleene, stable and well-founded fixpoints. In the context of logic programming, they found that Fitting's (three- or four-valued) immediate consequence operator is an approximating operator of the two-valued immediate consequence operator and that its four different types of fixpoints correspond exactly with the four major, equally named semantics of logic programs. They also identified approximating operators for DL and AEL and showed that the fixpoint theory induces all main and some new semantics in these fields (Denecker et al., 2003). Moreover, by showing that Konolige's mapping from DL to AEL preserves the approximating operator, AEL and DL were essentially unified and hence, they resolved an old research question regarding the nature of these two logics: AEL and DL are "just" two different dialects of autoepistemic reasoning (Denecker et al., 2003, 2011).

The study of these approximating operators, which is called approximation fixpoint theory, has continued. It is now commonly used to define semantics of extensions of logic programs, such as logic programs with aggregates (Pelov et al., 2007) and HEX logic programs (Antic et al., 2013). Truszczynski (2006) defined algebraic generalisations of strong and uniform equivalence using AFT. Vennekens et al. (2006) used AFT in an algebraical modularity study for logic programming, AEL and DL. Recently, Strass (2013) showed that many semantics from Dung's argumentation frameworks (AFs) and abstract dialectical frameworks can be obtained by direct applications of AFT. Bi et al. (2014) extended AFT with approximators allowing for inconsistencies and used it to integrate description logics with logic programs. Bogaerts et al. (2014c) defined the causal logic FO(C) as an instantiation of AFT.

1.5 Groundedness

In this dissertation, we extend AFT with several concepts. In Chapter 3, we extend AFT with the notion of a grounded fixpoint, a fixpoint closely related to stable fixpoints with similar properties, but that is completely determined by the operator, not by the choice of an approximator. We apply this theory to different research domains.

- In the context of logic programming, our theory yields an intuitive, purely two-valued semantics that is easily extensible and that formalises well-known intuitions related to unfounded sets.
- We show that two of the main semantics of AFs can be characterised as grounded fixpoints of previously defined operators and discuss grounded fixpoints in the context of ADFs.
- Applied to autoepistemic logic and default logic, groundedness formalises intuitions described by Konolige (1988).

In Chapter 4, we continue the study of groundedness. In that chapter, we generalise groundedness to a partial context: we define A-grounded bilattice points for an approximator A of O. We show this extended notion coincides with groundedness for exact lattice points. We also show that all A-stable fixpoints are A-grounded and provide a novel characterisation of the A-well-founded fixpoint in terms of A-groundedness. We apply this theory to logic programming.

In Chapter 5, we expose and solve a problem with the well-founded semantics for AEL. We show that for a class of autoepistemic theories, the well-founded semantics fails to identify the intended model. We provide an algebraical generalisation of this class of theories, namely *locally monotone lattice operators*. We give, in general, a refinement of the well-founded semantics which works on sets of lattice elements instead of intervals and show that for locally monotone operators, this refinement yields a unique fixpoint. Our refinement of the well-founded semantics is based on groundedness.

1.6 Contributions to Approximation Fixpoint Theory

The main contributions of the research presented in this dissertations are as follows.

- We define grounded lattice points and grounded bilattice points and discuss the relationship with other fixpoints studied in AFT.
- We find a new characterisation of the A-well-founded fixpoint as the least precise A-grounded fixpoint.
- We discuss the meaning of groundedness in logic programming, autoepistemic logic, default logic, AFs and ADFs; we show that in these contexts groundedness often formalises existing intuitions.

- We define a class of autoepistemic theories with a clear intended model and show that the well-founded semantics fails to identify this model. We generalise this observation to the algebraical setting, resulting in the class of locally monotone lattice operators.
- We define, algebraically, a refined version of the Kripke-Kleene and the well-founded semantics and show that the latter semantics, applied to AEL, succeeds to identify the intended model for the aforementioned class of autoepistemic theories.

Chapter 2

Preliminaries

2.1 Lattices and Operators

A *partially ordered set (poset)* $\langle L, \leq \rangle$ is a set L equipped with a partial order $\leq$, i.e., a reflexive, antisymmetric, transitive relation. As usual, we write $x < y$ as abbreviation for $x \leq y \wedge x \neq y$. If S is a subset of L, then x is an *upper bound*, respectively a *lower bound* of S if for every $s \in S$, it holds that $s \leq x$ respectively $x \leq s$. An element x is a *least upper bound*, respectively *greatest lower bound* of S if it is an upper bound that is smaller than or equal to every other upper bound, respectively a lower bound that is greater than every other lower bound. If S has a least upper bound, respectively a greatest lower bound, we denote it $\mathrm{lub}(S)$, respectively $\mathrm{glb}(S)$. As is customary, we sometimes call a greatest lower bound a *meet*, and a least upper bound a *join* and use the related notations $\bigwedge S = \mathrm{glb}(S)$, $x \wedge y = \mathrm{glb}(\{x, y\})$, $\bigvee S = \mathrm{lub}(S)$ and $x \vee y = \mathrm{lub}(\{x, y\})$. We call $\langle L, \leq \rangle$ a *bounded lattice* if every finite subset of L has a least upper bound and a greatest lower bound. We call $\langle L, \leq \rangle$ a *complete lattice* if every subset of L has a least upper bound and a greatest lower bound. A complete lattice has both a least element $\bot$ and a greatest element $\top$.

A lattice is *distributive* if $\wedge$ and $\vee$ distribute over each other, a bounded lattice is *complemented* if every element $x \in L$ has a *complement*: an element $\neg x \in L$ satisfying $x \wedge \neg x = \bot$ and $x \vee \neg x = \top$. A *Boolean lattice* is a distributive complemented lattice.

An operator $O : L \to L$ is *monotone* if $x \leq y$ implies that $O(x) \leq O(y)$ and *anti-monotone* if $x \leq y$ implies that $O(y) \leq O(x)$. An element $x \in L$ is a *prefixpoint*, a *fixpoint*, a *postfixpoint* of O if $O(x) \leq x$, respectively $O(x) = x$,

$x \leq O(x)$. Every monotone operator O in a complete lattice has a least fixpoint, denoted $\text{lfp}(O)$, which is also O's least prefixpoint.

Definition 2.1.1. If O is a monotone operator, a *monotone induction* of O is a (possibly transfinite) sequence $(x_i)_{i \leq \alpha}$ such that

- $x_0 = \bot$,
- $x_i \leq x_{i+1} \leq O(x_i)$,
- $x_\lambda = \text{lub}(\{x_i \mid i < \lambda\})$, for limit ordinals $\lambda \leq \alpha$.

A monotone induction is *terminal* if there exists no $x_{\alpha+1} \neq x_\alpha$ such that $(x_i)_{i \leq \alpha+1}$ is a monotone induction.

All terminal monotone inductions of O converge to $\text{lfp}(O)$.

2.2 Approximation Fixpoint Theory

Given a lattice L, approximation fixpoint theory makes use of the bilattice L^2. We define two *projection* functions for pairs as usual: $(x, y)_1 = x$ and $(x, y)_2 = y$. Pairs $(x, y) \in L^2$ are used to approximate all elements in the interval $[x, y] = \{z \mid x \leq z \wedge z \leq y\}$. We call $(x, y) \in L^2$ *consistent* if $x \leq y$, that is, if $[x, y]$ is non-empty. We use L^c to denote the set of consistent elements. Elements $(x, x) \in L^c$ are called *exact*; they constitute the embedding of L in L^2. We sometimes abuse notation and use the tuple (x, y) and the interval $[x, y]$ interchangeably. The *precision ordering* on L^2 is defined as $(x, y) \leq_p (u, v)$ if $x \leq u$ and $v \leq y$. In case (u, v) is consistent, this means that (x, y) approximates all elements approximated by (u, v), or in other words that $[u, v] \subseteq [x, y]$. If L is a complete lattice, then $\langle L^2, \leq_p \rangle$ is also a complete lattice.

AFT studies fixpoints of lattice operators $O : L \to L$ through operators approximating O. An operator $A : L^2 \to L^2$ is an *approximator* of O if it is $\leq_p$-monotone, and has the property that for all x, $O(x) \in A(x, x)$. Approximators are internal in L^c (i.e., map L^c into L^c). As usual, we often restrict our attention to *symmetric* approximators: approximators A such that for all x and y, $A(x, y)_1 = A(y, x)_2$. DMT (2004) showed that the consistent fixpoints of interest (supported, stable, well-founded) are uniquely determined by an approximator's restriction to L^c, hence, sometimes we only define approximators on L^c.

AFT studies fixpoints of O using fixpoints of A.

- The *A-Kripke-Kleene fixpoint* is the $\leq_p$-least fixpoint of A and has the property that it approximates all fixpoints of O.
- A *partial A-stable fixpoint* is a pair (x, y) such that $x = \text{lfp}(A(\cdot, y)_1)$ and $y = \text{lfp}(A(x, \cdot)_2)$, where $A(\cdot, y)_1$ denotes the operator $L \to L : x \mapsto A(x, y)_1$ and analogously for $A(x, \cdot)_2$.
- The *A-well-founded fixpoint* is the least precise partial A-stable fixpoint.
- An *A-stable fixpoint* of O is a fixpoint x of O such that (x, x) is a partial A-stable fixpoint. This is equivalent with the condition that $x = \text{lfp}(A(\cdot, x)_1)$.

The A-Kripke-Kleene fixpoint of O can be constructed as the limit of any monotone induction of A. For the A-well-founded fixpoint, a similar constructive characterisation has been worked out by Denecker and Vennekens (2007):

Definition 2.2.1. An *A-refinement* of (x, y) is a pair $(x', y') \in L^2$ satisfying one of the following two conditions:

1. $(x, y) \leq_p (x', y') \leq_p A(x, y)$, or
2. $x' = x$ and $A(x, y')_2 \leq y' \leq y$.

An A-refinement is *strict* if $(x, y) \neq (x', y')$.

We call the first type (1.) of refinements *application refinements* and the second type (2.) *unfoundedness refinements*. If (x', y') is an A-refinement of (x, y) and A is clear from the concepts, we often denote it $(x, y) \to (x', y')$.

Definition 2.2.2. A *well-founded induction* of A is a sequence $(x_i, y_i)_{i \leq \beta}$ with β an ordinal such that

- $(x_0, y_0) = (\bot, \top)$;
- (x_{i+1}, y_{i+1}) is an A-refinement of (x_i, y_i), for all $i < \beta$;
- $(x_\lambda, y_\lambda) = \text{lub}_{\leq_p} \{(x_i, y_i) \mid i < \lambda\}$ for each limit ordinal $\lambda \leq \beta$.

A well-founded induction is *terminal* if its limit (x_β, y_β) has no strict A-refinements.

A well-founded induction is an algebraical generalisation of the well-founded model construction defined by Van Gelder et al. (1991). The first type of refinement corresponds to making a partial structure more precise by applying

Fitting's immediate consequence operator; the second type of refinement corresponds to making a structure more precise by eliminating an unfounded set.

For a given approximator A, there are many different terminal well-founded inductions of A. Denecker and Vennekens (2007) showed that they all have the same limit, which equals the A-well-founded fixpoint of O. Furthermore, if A is symmetric, the A-well-founded fixpoint of O (and in fact, every tuple in a well-founded induction of A) is consistent.

The precision order can be pointwise extended to the family of approximators of O. It then follows that more precise approximators have a more precise well-founded fixpoint and that they have more stable fixpoints. DMT (2004) showed that there exists a most precise approximator, U_O, called the ultimate approximator of O. This operator is defined by

$$U_O : L^c \to L^c : (x, y) \mapsto (\bigwedge O([x, y]), \bigvee O([x, y])).$$

Here, we used the notation $O(X) = \{O(x) \mid x \in X\}$ for a set $X \subseteq L$. It then follows that for every approximator A, all A-stable fixpoints are U_O-stable fixpoints, and the U_O-well-founded fixpoint is always more precise than the A-well-founded fixpoint. We refer to U_O-stable fixpoints as *ultimate stable fixpoints* of O and to the U_O-well-founded fixpoint as the *ultimate well-founded fixpoint* of O. Semantics defined using the ultimate approximator have as advantage that they only depend on O since the approximator can be derived from O.

2.3 Logic Programming

We often illustrate our abstract results in the context of logic programming. We recall some preliminaries. We restrict ourselves to propositional logic programs, but allow arbitrary propositional formulas in rule bodies. However, our results basically apply to all extensions of logic programming that admit an immediate consequence operator (non-propositional ones, aggregates in the body, etc.).

Let Σ be a propositional alphabet, i.e., a collection of symbols which are called *atoms*. A *literal* is an atom p or the negation $\neg q$ of an atom q. A logic program $\mathcal{P}$ is a set of *rules* r of the form $h \leftarrow \varphi$, where h is an atom called the *head* of r, denoted $head(r)$, and φ is a propositional formula called the *body* of r, denoted $body(r)$. An interpretation I of the alphabet Σ is an element of 2^Σ, i.e., a subset of Σ. The set of interpretations 2^Σ forms a lattice equipped with the order $\subseteq$.

$A \wedge B$		B	
		t	**f**
A	**t**	**t**	**f**
	f	**f**	**f**

$A \vee B$		B	
		t	**f**
A	**t**	**t**	**t**
	f	**t**	**f**

		$\neg A$
	t	**f**
A	**f**	**t**

Figure 2.1: The truth tables for propositional logic.

The truth value (**t** or **f**) of a propositional formula φ in a structure I, denoted φ^I, is defined as usual based on the standard truth tables for propositional logic (see Figure 2.1).

With a logic program $\mathcal{P}$, we associate an immediate consequence operator (van Emden and Kowalski, 1976) $T_\mathcal{P}$ that maps a structure I to

$$T_\mathcal{P}(I) = \{p \mid \exists r \in \mathcal{P} : head(r) = p \wedge body(r)^I = \mathbf{t}\}.$$

The *supported models* of $\mathcal{P}$ are the fixpoints of $T_\mathcal{P}$.

In the context of logic programming, elements of the bilattice $\left(2^\Sigma\right)^2$ are four-valued interpretations, pairs $\mathcal{I} = (I_1, I_2)$ of interpretations. The pair (I_1, I_2) maps every atom to a tuple of two truth values; such a tuple corresponds to a four-valued truth value (true $(\mathbf{t}, \mathbf{t})$, false $(\mathbf{f}, \mathbf{f})$, unknown $(\mathbf{f}, \mathbf{t})$ or inconsistent $(\mathbf{t}, \mathbf{f})$). With this interpretation I_1 represents al atoms that are *certainly true* and I_2 represents all atoms that are *possibly true*. The pair (I_1, I_2) approximates all interpretations I' with $I_1 \subseteq I' \subseteq I_2$. We often identify an interpretation I with the four-valued interpretation (I, I). If $\mathcal{I} = (I_1, I_2)$ is a (four-valued) interpretation, and $U \subseteq \Sigma$, we write $\mathcal{I}[U : \mathbf{f}]$ for the (four-valued) interpretation that equals $\mathcal{I}$ on all elements not in U and that interprets all elements in U as **f**, i.e., the interpretation $(I_1 \setminus U, I_2 \setminus U)$. We are mostly concerned with consistent (also called partial or three-valued) interpretations: tuples $\mathcal{I} = (I_1, I_2)$ with $I_1 \subseteq I_2$. For such an interpretation, the atoms in I_1 are *true* (**t**) in $\mathcal{I}$, the atoms in $I_2 \setminus I_1$ are *unknown* (**u**) in $\mathcal{I}$ and the other atoms are *false* (**f**) in $\mathcal{I}$. If $\mathcal{I}$ is a three-valued interpretation, and φ a formula, we write $\varphi^\mathcal{I}$ for the standard three-valued valuation based on the Kleene truth tables (see Figure 2.2). An alternative valuation is the *supervaluation*; with this valuation, the value of a formula φ is **t** (respectively **f**) in partial interpretation $\mathcal{I}$ if and only if it is **t** (respectively **f**) in all interpretations approximated by $\mathcal{I}$; it is unknown otherwise.

We call two formulas 3-*equivalent* if they have the same truth value in all three-valued interpretations and 2-*equivalent* if they have the same truth value in all (two-valued) interpretations. Several approximators have been defined for logic programs. The most common is Fitting's immediate consequence operator $\Psi_\mathcal{P}$ (Fitting, 2002), a direct generalisation of $T_\mathcal{P}$ to partial interpretations defined

$A \wedge B$		B		
		t	**f**	**u**
A	**t**	**t**	**f**	**u**
	f	**f**	**f**	**f**
	u	**u**	**f**	**u**

$A \vee B$		B		
		t	**f**	**u**
A	**t**	**t**	**t**	**t**
	f	**t**	**f**	**u**
	u	**t**	**u**	**u**

		$\neg A$
A	**t**	**f**
	f	**t**
	u	**u**

Figure 2.2: The Kleene truth tables (Kleene, 1938).

by

$$\Psi_{\mathcal{P}}(\mathcal{I})_1 = \{a \in \Sigma \mid body(r)^{\mathcal{I}} = \mathbf{t} \text{ for some rule } r \in \mathcal{P} \text{ with } head(r) = a\},$$

$$\Psi_{\mathcal{P}}(\mathcal{I})_2 = \{a \in \Sigma \mid body(r)^{\mathcal{I}} \neq \mathbf{f} \text{ for some rule } r \in \mathcal{P} \text{ with } head(r) = a\}.$$

DMT (2000) showed that the well-founded fixpoint of $\Psi_{\mathcal{P}}$ is the well-founded model of $\mathcal{P}$ as defined by Van Gelder et al. (1991) and that $\Psi_{\mathcal{P}}$-stable fixpoints are exactly the stable models of $\mathcal{P}$ as defined by Gelfond and Lifschitz (1988). In this case, the operator $\Psi_{\mathcal{P}}(\cdot, y)_1$ coincides with the immediate consequence operator of the Gelfond-Lifschitz reduct (Gelfond and Lifschitz, 1988). The most precise approximator of $T_{\mathcal{P}}$ is the ultimate approximator $U_{\mathcal{P}}$.

Replacing the body of a rule by a 3-equivalent formula obviously preserves $\Psi_{\mathcal{P}}$; replacing the body of a rule by a 2-equivalent formula preserves $T_{\mathcal{P}}$ and hence also $U_{\mathcal{P}}$. Thus, transformations that preserve 3-equivalence, preserve standard Kripke-Kleene, stable and well-founded semantics, and transformations preserving 2-equivalence preserve all ultimate semantics (ultimate Kripke-Kleene, ultimate stable, ultimate well-founded). The ease with which this can be proven demonstrates the power of AFT.

Preserving 2-equivalence is not enough to preserve standard semantics. For example, consider programs $\mathcal{P} = \{p \leftarrow p \vee \neg p\}$ and $\mathcal{P}' = \{p.\}$. Even though the body of the rule defining p in $\mathcal{P}$ is a tautology, $\{p\}$ is not a stable model of $\mathcal{P}$ while it is a stable model of $\mathcal{P}'$. But ultimate semantics treat these two programs identically. For instance, $\{p\}$ is the unique ultimate stable model of both programs.

While substituting formulas for 2-equivalent formulas in rule bodies preserves the ultimate but not necessarily the standard versions of semantics, a weaker equivalence property can still be guaranteed. It holds that the standard Kripke-Kleene and well-founded models of both programs are compatible with each other: no atom is true in the model of one and false in the model of the other program. This follows from the fact that both the ultimate Kripke-Kleene and the ultimate well-founded model are preserved by these substitutions and that

they are consistent and more precise than the standard Kripke-Kleene and well-founded model respectively.

The nice property that ultimate semantics only depend on the operator comes at a cost. DMT (2004) showed that deciding whether $\mathcal{P}$ has an ultimate stable model is Σ_P^2-complete, while that same task is only NP-complete for classical stable models.

Chapter 3

Grounded fixpoints and their applications in knowledge representation

The contents of this chapter are accepted for publication in *Artificial Intelligence* (Bogaerts et al., 2015b). A short version was published in the proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI'15) (Bogaerts et al., 2015a).

3.1 Introduction

In this chapter, we formally use fixpoint theory to investigate an intuition that is found in all logic domains in which approximation fixpoint theory is applied. In those domains, researchers have made use of a similar intuition: that facts (or models) can be derived *from the ground up*. They typically phrase this intuition by saying that, e.g., the facts should be *grounded*, or that they should not be *unfounded*, or that they should be supported by *cycle-free* arguments, or by arguments that contain no vicious circles, et cetera. In several cases, great efforts were made to refine semantics which did allow ungrounded models or facts. For example, it is well-known that the completion semantics of logic programs allows ungrounded models, e.g., for the transitive closure program. The efforts to avoid these led to the development of perfect, stable and well-founded semantics. Also for AEL, it was known that Moore's expansion

semantics accepted ungrounded models, e.g., for the theory $\{KP \Rightarrow P\}$ which has the ungrounded model in which P is known but this knowledge is self-supporting. Examples like this motivated several attempts to refine Moore's semantics, among others by Halpern and Moses (1985), Konolige (1988) and Niemelä (1991).

We formalise the intuition of groundedness in the context of algebraical fixpoint theory. We call a lattice element $x \in L$ *grounded* for lattice operator $O : L \to L$ if for all $v \in L$ such that $O(x \wedge v) \leq v$, it holds that $x \leq v$. We investigate this notion on the algebraical level in AFT and on the logical level in the context of logic programming, autoepistemic logic, default logic and abstract argumentation frameworks. We explain what grounded points and fixpoints mean in these logics. We use groundedness as an *ordering principle*. Within each of the application domains, we order semantics into two categories: grounded and ungrounded semantics, where we call a semantics grounded if all its models are grounded. Over the different domains, we bring order by unifying intuitions that existed in each of these fields. We investigate a novel semantics for the various logics that is based on grounded fixpoints. Our results unveil a remarkable uniformity in intuition and mathematics in these fields and lead to a new candidate semantics with some very appealing properties, not in the least the mathematical simplicity and generality to define it in the context of operator-based logics and logic constructs.

We can summarise the main contributions of this chapter as follows. We extend AFT with the notion of a grounded fixpoint, a fixpoint closely related to stable and well-founded fixpoints. We show that if the Kripke-Kleene fixpoint is exact, then it is grounded. If the well-founded fixpoint is exact, then it is the unique grounded and the unique stable fixpoint. Otherwise, stable fixpoints are grounded but not necessarily the other way around. A useful feature of grounded fixpoints that distinguishes them from stable and well-founded fixpoints is that they are determined by O and do not require the choice of an approximator. We then apply this theory to different logical research domains. In all domains, we explain the meaning of grounded fixpoints, relate them to attempts to formalise groundedness, study which semantics are grounded and finally, we explore the semantics induced by grounded fixpoints.

- In the context of logic programming, our theory yields an intuitive, purely two-valued, semantics that is easily extensible and that formalises well-known intuitions related to unfounded sets.
- We show that two of the main semantics of argumentation frameworks (AFs) can be characterised as grounded fixpoints of previously defined operators and discuss grounded fixpoints in the context of abstract dialectical frameworks (ADFs).

- Applied to autoepistemic logic and default logic, groundedness turns out to provide an alternative and improved formalisation of intuitions described by Konolige (1988).

3.2 Grounded Fixpoints

Let $\langle L, \leq \rangle$ be a complete lattice and $O : L \to L$ a lattice operator, fixed throughout this entire section. We start by giving the central definition of this chapter, namely the notion of groundedness.

Definition 3.2.1 (Grounded)**.** We call $x \in L$ *grounded* for O if for each $v \in L$ such that $O(x \wedge v) \leq v$, it holds that $x \leq v$. We call x a *grounded fixpoint* of O if it is a fixpoint of O and it is grounded for O.

This concept is strongly related to the following.

Definition 3.2.2 (Strictly grounded)**.** We call $x \in L$ *strictly grounded* for O if there is no $y \in L$ such that $y < x$ and $(O(y) \wedge x) \leq y$.

The intuition behind these concepts is very similar and is easy to explain if we assume that the elements of L are sets of "facts" of some kind and the $\leq$ relation is the subset relation between such sets. In this case, $\wedge$ is the intersection and $\vee$ the union of sets. Intuitively, a set of facts x is (strictly) grounded if it can be "built from the ground up" by O. Such sets are built in several stages. Facts that are derived in later stages depend on those of earlier stages. This means that x has a stratified internal structure (it contains facts from different stages). If we remove multiple stages from x, we cannot expect that with one application of O, all of the removed facts will be reconstructed, but we should expect that at least some of the removed facts of x reappear, in particular those in the lowest stage from where facts were deleted. This idea is formalised in slightly different ways in the two definitions.

If L is a powerset lattice, this intuition directly translates to $\forall u \neq \emptyset : u \subseteq x \Rightarrow O(x \setminus u) \cap u \neq \emptyset$, i.e., whenever a set of elements u is removed from x, at least one of these elements returns. To express it in the context of lattices in general, the statement needs to be reformulated without using set subtraction. There are two set-theoretically equivalent ways to do this.

In the definition of *strictly grounded* point, removing strata from x corresponds to taking $y < x$ (y represents $x \setminus u$ in this case). The condition that no elements of u come back corresponds to $(O(y) \wedge x) \leq y$ (all elements of $O(y) \cap x$ are in y; hence not in u).

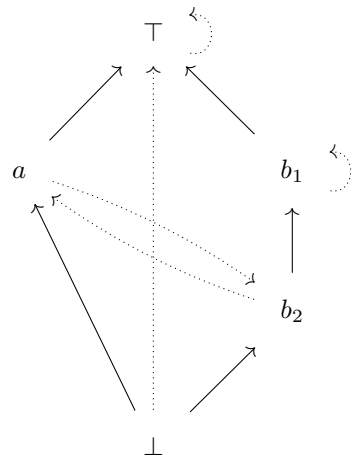
In the definition of *grounded* point, a slightly different but equivalent set-theoretic expression is used as a starting point: $\forall u : O(x \setminus u) \cap u = \emptyset \Rightarrow u \cap x = \emptyset$. Removing strata from x corresponds to selecting a $v \in L$ and taking $x \wedge v$ (v corresponds to the complement of u, hence $x \setminus u$ to $x \wedge v$ and the fact that $u \cap x = \emptyset$ corresponds to $x \leq v$). The condition that $O(x \wedge v)$ reintroduces no elements from u corresponds to $O(x \wedge v) \leq v$; for a point to be grounded, if $O(x \wedge v)$ reintroduces no elements, then $u \cap x$ must be empty, i.e., no points were removed in the first place.

In what follows, we study some properties of (strictly) grounded (fix)points. We start by showing the tight relationship between the two concepts. In general, every strictly grounded point is grounded but not necessarily the other way around. Unsurprisingly, in the context of powerset lattices — the context in which we explained the intuitions — the two notions coincide.

Proposition 3.2.3. *If x is strictly grounded for O, then x is grounded for O.*

Proof. Assume x is strictly grounded and $v \in L$ is such that $O(x \wedge v) \leq v$. Let y denote $x \wedge v$. Then $O(y) \leq v$, and hence $(O(y) \wedge x) \leq (x \wedge v) = y$. Since $(O(y) \wedge x) \leq y$ and x is strictly grounded, it cannot be the case that $y < x$. Since $(x \wedge v) = y \leq x$, equality holds, i.e. $(x \wedge v) = x$ and $x \leq v$. This shows that x is indeed grounded. □

Example 3.2.4. The converse of Proposition 3.2.3 does not hold. Consider the lattice and the operator represented by the following graph, where full edges express the order relation (to be precise, the $\leq$ relation is the reflexive transitive closure of these edges) and the dotted edges represent the operator:



In this case, b_1 is grounded but not strictly grounded, as can be seen by taking $y = b_2$.

Proposition 3.2.5. *If L is a Boolean lattice, then a point $x \in L$ is grounded if and only if it is strictly grounded.*

Proof. We recall that in the context of powerset lattices the greatest lower bound is the intersection and least upper bound is the union.

Proposition 3.2.3 guarantees that we only need to show that all grounded points are strictly grounded. Hence, suppose x is grounded. Assume towards contradiction that x is not strictly grounded, i.e., that for some $y < x$, $O(y) \wedge x \leq y$. Take $v = y \vee \neg x$. It holds that

$$\begin{aligned} v \wedge x &= (y \vee \neg x) \wedge x \\ &= (y \wedge x) \vee (\neg x \wedge x) \\ &= y \wedge x \\ &= y, \end{aligned}$$

since $y < x$. Since $x \neq y$, it must hold that $x \not\leq v$ (otherwise, $x = x \wedge v = y$). Since we assumed that $O(y) \wedge x \leq y$, it also holds that

$$\begin{aligned} v &= y \vee \neg x \\ &\geq (O(y) \wedge x) \vee \neg x \\ &= O(y) \vee \neg x. \end{aligned}$$

Hence $O(y) \vee \neg x \leq v$ and also $O(x \wedge v) = O(y) \leq v$. Together with the observation that $x \not\leq v$, we find a contradiction with the assumption that x is grounded. □

Corollary 3.2.6. *Let L be a powerset lattice $\langle 2^{\top}, \subseteq \rangle$. Then a point $x \in L$ is grounded if and only if it is strictly grounded.*

Proof. Follows immediately from the fact that powerset lattices are Boolean lattices. □

Proposition 3.2.7. *Let O be a monotone operator. If x is grounded for O then x is a postfixpoint of O that is less than or equal to* $\mathrm{lfp}(O)$*, i.e., $x \leq O(x)$ and $x \leq \mathrm{lfp}(O)$.*

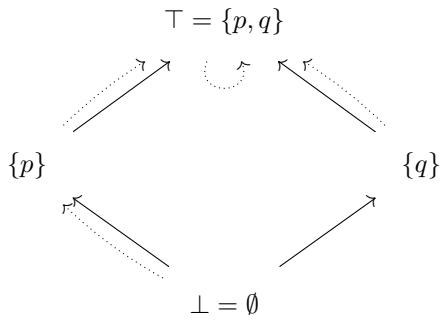
Proof. First, we show that $x \leq \mathrm{lfp}(O)$. Since O is monotone, we have $O(\mathrm{lfp}(O) \wedge x) \leq O(\mathrm{lfp}(O)) = \mathrm{lfp}(O)$. Hence, groundedness of x with $v = \mathrm{lfp}(O)$ indeed yields that $x \leq \mathrm{lfp}(O)$.

In order to show that x is a postfixpoint of O, take $v = O(x)$. Again using monotonicity of O, we find $O(v \wedge x) \leq O(x) = v$. Hence, groundedness yields that $x \leq v = O(x)$, and indeed x is a postfixpoint of O. □

Example 3.2.8. The converse of Proposition 3.2.7 does not hold. Consider the following logic program $\mathcal{P}$:

$$\left\{ \begin{array}{l} p. \\ q \leftarrow p \vee q. \end{array} \right\}$$

Its immediate consequence operator $T_{\mathcal{P}}$ is represented by the following graph:



$T_{\mathcal{P}}$ is a monotone operator with least fixpoint $\top$. Also, $\{q\}$ is a postfixpoint of $T_{\mathcal{P}}$ since $T_{\mathcal{P}}(\{q\}) = \top \geq \{q\}$. However, $\{q\}$ is not grounded since $T_{\mathcal{P}}(\{q\} \wedge \{p\}) = T_{\mathcal{P}}(\bot) = \{p\}$, while $\{q\} \not\leq \{p\}$.

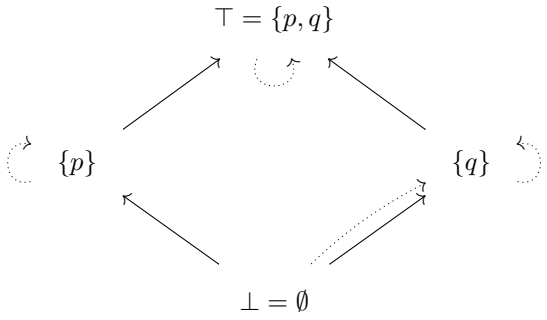
Proposition 3.2.9. *All grounded fixpoints of O are minimal fixpoints of O.*

Proof. Suppose x is grounded and y and x are fixpoints of O with $y \leq x$. In this case, $O(x \wedge y) = O(y) = y \leq y$. Thus, since x is grounded, we conclude that $x \leq y$, which yields $x = y$. We find that indeed, all grounded fixpoints are minimal fixpoints. □

Example 3.2.10. The converse of Proposition 3.2.9 does not hold. Consider the logic program $\mathcal{P}$:

$$\left\{ \begin{array}{l} p \leftarrow p. \\ q \leftarrow \neg p \vee q. \end{array} \right\}$$

This logic program has as immediate consequence operator $T_{\mathcal{P}}$:



In this case, $\{p\}$ is a minimal fixpoint of $T_{\mathcal{P}}$, but $\{p\}$ is not grounded since $T_{\mathcal{P}}(\{p\} \wedge \{q\}) = T_{\mathcal{P}}(\bot) = \{q\}$, while $\{p\} \not\leq \{q\}$. On the other hand $\{q\}$ is grounded for $T_{\mathcal{P}}$ since $O(\{q\} \wedge v) = \{q\}$ for all $v \in L$. Hence for $v \in \{\bot, \{p\}\}$, it holds that $O(x \wedge v) \not\leq v$ and for all other $v \in L$ it holds that $\{q\} \leq v$.

Proposition 3.2.11. *A monotone operator has exactly one grounded (and strictly grounded) fixpoint, namely its least fixpoint.*

Proof. Proposition 3.2.9 guarantees that grounded fixpoints are minimal, hence a monotone operator O can have at most one grounded fixpoint $\mathrm{lfp}(O)$. Now we show that $\mathrm{lfp}(O)$ is indeed strictly grounded. Since O is monotone, for every $y \leq \mathrm{lfp}(O)$, it holds that $O(y) \leq O(\mathrm{lfp}(O)) = \mathrm{lfp}(O)$ and hence that $O(y) \wedge \mathrm{lfp}(O) = O(y)$. Now suppose that for some $y \leq \mathrm{lfp}(O)$, $O(y) \wedge \mathrm{lfp}(O) \leq y$. Then by the previous also $O(y) \leq y$. Thus y is a prefixpoint of O. However, $\mathrm{lfp}(O)$ is the least prefixpoint of O, hence $\mathrm{lfp}(O) \leq y$, and thus $y = \mathrm{lfp}(O)$. We conclude that $\mathrm{lfp}(O)$ is indeed strictly grounded. From Proposition 3.2.3, it then follows that x is grounded as well. □

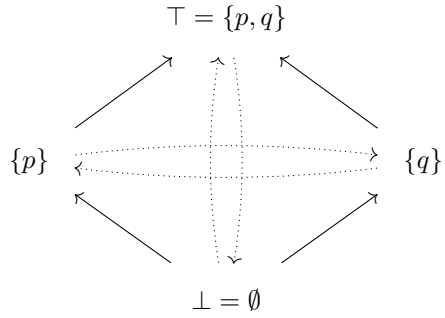
Proposition 3.2.12. *Every postfixpoint of an anti-monotone operator is strictly grounded.*

Proof. Suppose x is a postfixpoint of an anti-monotone operator O, i.e., $x \leq O(x)$ and that $y \leq x$. In that case $O(y) \geq O(x) \geq x$. If $O(y) \wedge x \leq y$, then it follows that $x \leq y$. Thus indeed $x = y$ and x is strictly grounded. □

Example 3.2.13. The converse of Proposition 3.2.12 does not hold. Consider the logic program $\mathcal{P}$:

$$\left\{ \begin{array}{l} p \leftarrow \neg p. \\ q \leftarrow \neg q. \end{array} \right\}$$

This logic program has as immediate consequence operator $T_{\mathcal{P}}$:



The operator $T_{\mathcal{P}}$ is anti-monotone, and $\{p\}$ is strictly grounded for $T_{\mathcal{P}}$. However, $\{p\}$ is not a postfixpoint of $T_{\mathcal{P}}$.

3.3 Grounded Fixpoints and Approximation Fixpoint Theory

In this section, we discuss how groundedness relates to AFT. More concretely, we show that all (ultimate) stable fixpoints are grounded and that all grounded fixpoints are approximated by the (ultimate) well-founded fixpoint.

Proposition 3.3.1. *All ultimate stable fixpoints of O are (strictly) grounded fixpoints.*

Proof. Let x be an ultimate stable fixpoint of O. Thus, (x, x) is a fixpoint of the ultimate stable operator, i.e., $x = U_O(x)_1 = \text{lfp}(\bigwedge O([\cdot, x]))$. Since $O(x) = \bigwedge O([x, x])$, it follows that x is also a fixpoint of O. Now suppose for some $y \leq x$, $O(y) \wedge x \leq y$; we show that $x = y$. We know that

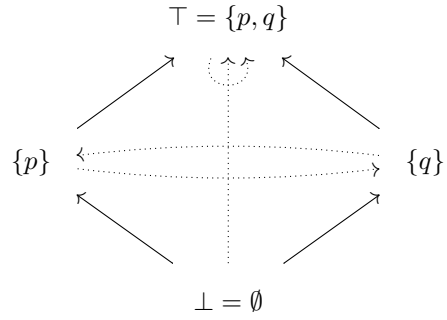
$$\bigwedge O([y, x]) \leq O(y) \wedge O(x) = O(y) \wedge x \leq y.$$

Thus, y is a prefixpoint of the monotone operator $\bigwedge O([\cdot, x])$. Since x is the least fixpoint (and also the least prefixpoint) of that same operator, we find that $x \leq y$, and thus $x = y$, which shows that x is strictly grounded indeed. □

Example 3.3.2. The converse of Proposition 3.3.1 does not always hold. Consider the logic program $\mathcal{P}$:

$$\left\{ \begin{array}{l} p \leftarrow \neg p \vee q. \\ q \leftarrow \neg q \vee p. \end{array} \right\}$$

This logic program has as immediate consequence operator $T_{\mathcal{P}}$:



$\top$ is grounded for $T_{\mathcal{P}}$, since the only v with $T_{\mathcal{P}}(\top \wedge v) = T_{\mathcal{P}}(v) \leq v$ is $\top$ itself. However, since $T_{\mathcal{P}}([\bot, \top]) = L \setminus \{\bot\}$ and $\{p\} \wedge \{q\} = \bot$, it follows that $\bigwedge(T_{\mathcal{P}}[\bot, \top]) = \bot$. Thus, $\mathrm{lfp}(\bigwedge T_{\mathcal{P}}([\cdot, \top])) = \bot$. Therefore, $\top$ is not an ultimate stable fixpoint of $T_{\mathcal{P}}$.

The fact that all A-stable fixpoints are ultimate stable fixpoints yields:

Corollary 3.3.3. *If A is an approximator of O, then all A-stable fixpoints are (strictly) grounded fixpoints of O.*

Theorem 3.3.4. *The well-founded fixpoint (u, v) of a symmetric approximator A of O approximates all grounded fixpoints of O.*

Proof. Let $(a_i, b_i)_{i \leq \beta}$ be a well-founded induction of A and let x be a grounded fixpoint of O. We show by induction that for every $i \leq \beta$, $a_i \leq x \leq b_i$. The result trivially holds for $i = 0$ since $(a_0, b_0) = (\bot, \top)$. It is also clear that the property is preserved in limit ordinals. Hence, all we need to show is that the property is preserved by A-refinements. Suppose (a', b') is an A-refinement of (a, b) and (a, b) approximates x. We show that also (a', b') approximates x, i.e., that $a' \leq x \leq b'$. We distinguish two cases.

First, assume that $(a, b) \leq_p (a', b') \leq_p A(a, b)$. Since x is a fixpoint of O and A an approximator of O, we find that $x = O(x) \in A(x, x) \subseteq A(a, b) \subseteq (a', b')$.

Second, assume that $a' = a$ and $A(a, b')_2 \leq b' \leq b$. Since every tuple in a well-founded induction of a symmetric approximator is consistent, we know that $b' \geq a$. Since also $x \geq a$, we see that $a \leq x \wedge b' \leq b'$. Hence $x \wedge b' \in [a, b']$, thus $O(x \wedge b') \in A(a, b')$, and we see that $O(x \wedge b') \leq A(a, b')_2 \leq b'$. Since x is grounded, this implies that $x \leq b'$; we conclude that also in this case $x \in [a', b']$.

We have thus shown that every step in a well-founded induction of A preserves all grounded fixpoints. □

Corollary 3.3.5. *If the well-founded fixpoint of a symmetric approximator A of O is exact, then this point is the unique (strictly) grounded fixpoint of O.*

Since the Kripke-Kleene fixpoint approximates the well-founded fixpoint, we also get the following property.

Corollary 3.3.6. *If the Kripke-Kleene fixpoint of a symmetric approximator A of O is exact, then it is the unique fixpoint of O and it is (strictly) grounded.*

3.4 Grounded Fixpoints of Logic Programs

In this section, we discuss grounded fixpoints in the context of logic programming. It follows immediately from the algebraical results (Corollary 3.3.3 and Theorem 3.3.4) that stable models are grounded fixpoints of the immediate consequence operator and that all grounded fixpoints are minimal fixpoints approximated by the well-founded model. Furthermore, if the Kripke-Kleene or well-founded model is exact, then it is the unique grounded fixpoint of $T_{\mathcal{P}}$.

Grounded fixpoints can be explained in terms of unfounded sets, a notion that was first defined by Van Gelder, Ross and Schlipf (1991) in their seminal paper introducing the well-founded semantics. Unfounded sets of a three-valued interpretation are a key concept in the construction of the well-founded model. Intuitively, an unfounded set is a set of atoms that might circularly support themselves, but have no support from outside. Stated differently, an unfounded set of a logic program $\mathcal{P}$ with respect to a (partial) interpretation $\mathcal{I}$ is a set U of atoms such that $\mathcal{P}$ does not provide support for any atom in U if the atoms in U are assumed false.

Below, we define the concept of unfounded set in the context of two-valued interpretations. For clarity, we refer to our unfounded sets as "2-unfounded sets" and to the original definition by Van Gelder, Ross and Schlipf as "GRS-unfounded sets".

Definition 3.4.1 (2-Unfounded set)**.** Let $\mathcal{P}$ be a logic program and $I \subseteq \Sigma$ an interpretation.

A set $U \subseteq \Sigma$ is a *2-unfounded set* of I (with respect to $\mathcal{P}$) if for each rule $r \in \mathcal{P}$ with $head(r) \in U$, $body(r)^{I[U:\mathbf{f}]}$ is false. A 2-unfounded set U of I is called *proper* if U is a nonempty subset of I.

Thus, U is a 2-unfounded set of I if after revising I by setting atoms of U to false, no atom in U can be derived.

All interpretations I admit 2-unfounded sets, in particular the empty set $\emptyset$ and every set U consisting of atoms p that are false in I and for which every rule $r \in \mathcal{P}$ with $head(r) = p$ has a false body in I. Indeed, for such sets, it holds that $I[U : \mathbf{f}] = I$ and no rule derives an element of U. However, not every interpretation I admits a proper 2-unfounded set. If I has a proper 2-unfounded set, then this means that I cannot be built up from the ground.

In Section 3.4.1, we investigate the relationship between this formalisation of unfounded sets and the original one. In particular, we extend the above definition to three-valued interpretations and show that the different notions of unfounded set are equivalent in the context of the well-founded model construction. First, we show how unfounded sets are related to the algebraical notion of groundedness.

The definition of 2-unfounded set can be easily rephrased in terms of the operator $T_{\mathcal{P}}$, as shown in the following proposition.

Proposition 3.4.2. *U is a 2-unfounded set of I with respect to $\mathcal{P}$ if and only if $T_{\mathcal{P}}(I[U : \mathbf{f}]) \cap U = \emptyset$.*

Proof. Follows immediately from the fact that $p \in T_{\mathcal{P}}(I[U : \mathbf{f}])$ if and only if for some rule $r \in \mathcal{P}$ with $head(r) = p$, it holds that $body(r)^{I[U:\mathbf{f}]} = \mathbf{t}$. □

Example 3.4.3. Let $\mathcal{P}$ be the following program:

$$\left\{ \begin{array}{l} p \leftarrow q. \\ q \leftarrow p. \\ r \leftarrow \neg p. \end{array} \right\}$$

Let I be the interpretation $\{p, q\}$. Then $U_1 = \{p, q\}$ is a 2-unfounded set of I since $I[U_1 : \mathbf{f}] = \{r\}$ and in this structure, the bodies of rules defining p and q are false. Alternatively, we notice that $T_{\mathcal{P}}(I[U_1 : \mathbf{f}]) \cap U_1 = \emptyset \cap U_1 = \emptyset$.

The set $U_2 = \{p\}$ is not a 2-unfounded set of I since the rule body for p evaluates to true in $I[U_2 : \mathbf{f}]$.

In what follows, we use $\overline{U}$ for the set complement of U, i.e., $\overline{U} = \Sigma \setminus U$.

Proposition 3.4.4. *Let $\mathcal{P}$ be a logic program, $I \in 2^{\Sigma}$ an interpretation, $U \subseteq \Sigma$. The following statements are equivalent:*

- *U is a 2-unfounded set of I,*
- *$T_{\mathcal{P}}(I \cap \overline{U}) \subseteq \overline{U}$, and*

- $T_{\mathcal{P}}(I \setminus U) \cap I \subseteq I \setminus U$.

Proof. The equivalence of the first two follows immediately from Proposition 3.4.2 since $I[U : \mathbf{f}] = I \setminus U = I \cap \overline{U}$ and for every set X, $X \subseteq \overline{U}$ if and only if $X \cap U = \emptyset$. The equivalence of the second and third statement follows from the fact that $I \setminus U = I \cap \overline{U}$ and for all subsets X, Y and Z of Σ it holds that $X \cap Y \subseteq Y \setminus Z$ if and only if $X \subseteq (\Sigma \setminus Z)$. □

Proposition 3.4.4 shows that U is a 2-unfounded set if and only if its complement satisfies the condition on v in Definition 3.2.1 if and only if $I \setminus U$ satisfies the condition on y in Definition 3.2.2. This allows us to reformulate the condition that I is grounded as follows.

Proposition 3.4.5. *An interpretation I is (strictly) grounded for $T_{\mathcal{P}}$ if and only if I does not contain atoms that belong to a 2-unfounded set U of I with respect to P.*

Proof. First, suppose I is grounded for $T_{\mathcal{P}}$ and U is a 2-unfounded set of I. Let $V = \overline{U}$ denote the set complement of U. Since U is a 2-unfounded set, $T_{\mathcal{P}}(I \cap V) \subseteq V$. Thus, the definition of groundedness yields $I \subseteq V$, and hence that $U \cap I = \emptyset$. We conclude that I is indeed disjoint from any 2-unfounded set.

The reverse direction is analogous. Suppose every 2-unfounded set is disjoint from I. Let V be such that $T_{\mathcal{P}}(I \cap V) \subseteq V$ and let $U = \overline{V}$ denote the complement of V. By Proposition 3.4.4 U is an unfounded set of I, hence it follows from our assumption that $I \cup U = \emptyset$ and hence $I \subseteq \overline{U} = V$.

We already established in Corollary 3.2.6 that groundedness is equivalent to strict groundedness in this context. □

Corollary 3.4.6. *A structure I is a grounded fixpoint of $T_{\mathcal{P}}$ if and only if it is a fixpoint of $T_{\mathcal{P}}$ and it has no proper 2-unfounded sets.*

We call grounded fixpoints of $T_{\mathcal{P}}$ *grounded models of* $\mathcal{P}$. Similarly to ultimate semantics, grounded models are insensitive to 2-equivalence-preserving rewritings in the bodies of rules: if $\mathcal{P}$ and $\mathcal{P}'$ are such that $T_{\mathcal{P}} = T_{\mathcal{P}'}$, then the grounded models of $\mathcal{P}$ and $\mathcal{P}'$ coincide. Also similar to ultimate semantics, the above property comes at a cost.

Theorem 3.4.7. *The problem "given a finite propositional logic program $\mathcal{P}$, decide whether $\mathcal{P}$ has a grounded model" is Σ_2^P-complete.*

The proof of this theorem is heavily inspired by the proof of a similar property for ultimate stable models (Theorem 6.12) by DMT (2004); we use the same reduction of a Σ_2^P-hard problem to our problem.

Proof. Given an interpretation I, any non-trivial 2-unfounded set of I is a witness that I is *not* grounded. Thus, verifying that I is not grounded is in NP and verifying that I is a grounded model is in co-NP. Thus the task of deciding whether there exists a grounded model certainly is in the class Σ_2^P.

We now show Σ_2^P-hardness of the problem of existence of a grounded model of a program $\mathcal{P}$. Let φ be a propositional formula in disjunctive normal form (DNF) over propositional symbols $x_1, \ldots, x_m, y_1, \ldots, y_n$. For an interpretation $I \subseteq \{x_1, \ldots, x_m\}$, we define φ_I as the formula obtained from φ by replacing all atoms $x_i \in I$ by $\mathbf{t}$ and all atoms $x_i \notin I$ by $\mathbf{f}$. Recall that the problem of deciding whether there exists an interpretation $I \subseteq \{x_1, \ldots, x_m\}$ such that φ_I is a tautology is Σ_2^P-hard. We now reduce this problem to our problem. For each x_i, we introduce a new variable x'_i; we will use x'_i to represent the negation of x_i. Let φ' be the formula obtained from φ by replacing all literals $\neg x_i$ by x'_i. We define a program $\mathcal{P}(\varphi)$ consisting of the following clauses

1. $x_i \leftarrow \neg x'_i$ and $x'_i \leftarrow \neg x_i$ for each $i \in \{1, \ldots, m\}$,
2. $y_i \leftarrow \varphi'$ for each $i \in \{1, \ldots, n\}$,
3. $p \leftarrow \varphi'$,
4. $q \leftarrow \neg p \wedge \neg q$.

We now show that there is an $I \subseteq \{x_1, \ldots, x_m\}$ such that φ_I is a tautology if and only if $\mathcal{P}(\varphi)$ has a grounded model. It is easy to see that in each fixpoint M of $T_{\mathcal{P}(\varphi)}$ the following properties hold:

1. q is false in M (if q is true $T_{\mathcal{P}(\varphi)}$ does not derive q),
2. p is true in M (otherwise $T_{\mathcal{P}(\varphi)}$ derives q),
3. $y_1, \ldots, y_n$ are true in M (since their rules have the same body as p),
4. for each $i \in \{1, \ldots, m\}$, exactly one of x_i and x'_i is true in M.

Given a set $I \subseteq \{x_1, \ldots, x_m\}$, we define $\check{I} = I \cup \{x'_i \mid x_i \notin I\}$. It follows from the above properties that for each fixpoint M of $T_{\mathcal{P}(\varphi)}$, there exists an I such that

$$M = \check{I} \cup \{p, y_1, \ldots, y_n\}.$$

Thus it suffices to show that if $I \subseteq \{x_1, \ldots, x_m\}$, then $M = \check{I} \cup \{p, y_1, \ldots, y_n\}$ is a grounded model of $\mathcal{P}(\varphi)$ if and only if φ_I is a tautology.

In order to prove this, we fix I and $M = \check{I} \cup \{p, y_1, \ldots, y_n\}$. Now, M is *not* a grounded model if and only if there exists a non-empty $U \subseteq M$ such that $T_{\mathcal{P}}(M \setminus U) \cap U = \emptyset$. By the structure of the rules defining x_i and x_i', each such U has the property that $x_i \notin U$ and $x_i' \notin U$. Hence $U \subseteq \{p, y_1, \ldots, y_n\}$. Since the bodies of rules defining atoms in U must be false in $M \setminus U$, such a U has the property that φ' is false in $M \setminus U$. But φ' is false in $M \setminus U$ if and only if φ_I is false in $\{y_1, \ldots, y_n\} \setminus U$. Thus, we conclude that M is *not* a grounded model if and only if there exists a truth assignment to $J \subseteq \{y_1, \ldots, y_n\}$ such that φ_I is false in J. Thus, M is *not* a grounded model if and only if φ_I is *not* a tautology, which is exactly what we needed to show. □

Let us briefly compare grounded model semantics with the two most frequently used semantics of logic programming: well-founded and stable semantics. First, we observe that these three semantics tend to prefer a subclass of the minimal models. We called this criterion groundedness and indeed found that stable semantics and two-valued well-founded semantics have the property that they only accept grounded interpretations.

Second, since these three semantics are closely related, it is to be expected that they often coincide. We established that for programs with a two-valued Kripke-Kleene model, the Kripke-Kleene semantics coincides with the supported model semantics and with the three semantics mentioned above. Also for programs with a two-valued well-founded model, the three semantics coincide. This sort of programs is common in applications for deductive databases (Datalog and extensions (Abiteboul and Vianu, 1991)) and for representing inductive definitions (Denecker and Ternovska, 2008; Denecker and Vennekens, 2014). In contrast, well-founded semantics only rarely coincides with stable semantics in the context of answer set programming. In the context where the well-founded model is three-valued, the question arises when stable and grounded models coincide. We illustrated in Example 3.3.2 that in this case, stable and grounded model semantics may disagree (since $\{p, q\}$ is not an ultimate stable model, it certainly is no stable model either). However, we observe that this example is quite extraordinary, and this is the case for all such programs that we found. It leads us to expect that both semantics coincide for large classes of answer set programming (ASP) programs. It is therefore an interesting topic for future research to search for characteristics of programs that guarantee that both semantics agree. If such properties can be identified, then within those classes the grounded model semantics gives an equivalent reformulation of the stable semantics. If these classes cover the pragmatically important classes of ASP programs (that is, if the ASP programs written for practical problems fall

inside them), then the grounded model semantics is an *elegant*, *intuitive* and *concise* variant of the standard stable semantics, which in practice coincides with it. And if pragmatically important classes of programs are discovered for which both semantics disagree, the question then is if other properties than groundedness can be identified that are possessed by stable but not by grounded models.

Several other semantics such as well-founded and stable semantics satisfy the desirable property of groundedness. However, unlike those semantics, grounded model semantics is *purely two-valued and algebraical*. The well-founded semantics explicitly uses three-valued interpretations in the well-founded model construction. Stable semantics uses three-valued logic implicitly in the sense that, as we showed, the Gelfond-Lifschitz reduct corresponds to an evaluation in a partial interpretation. One of the main advantages of grounded model semantics is that it is so easily definable for language extensions. All it takes is to define the (two-valued) immediate consequence operator. Typically this is quite easy (see the following paragraph). Note that the ultimate versions of the well-founded and stable semantics are purely algebraical as well but they are mathematically more involved since they still refer to three-valued interpretations (replacing Kleene valuation by supervaluation).

Grounded Fixpoints for Logic Programs with Abstract Constraint Atoms The fact that grounded model semantics is two-valued and algebraical makes it not only easier to understand, but also to *extend* the semantics. To illustrate this, we consider logic programs with abstract constraint atoms as defined by Marek et al. (2008). An *abstract constraint* is a collection $C \subseteq 2^{\Sigma}$. A *constraint atom* is an expression of the form $C(X)$, where $X \subseteq \Sigma$ and C is an abstract constraint. The goal of such an atom is to model constraints on subsets of X. The truth value of $C(X)$ in interpretation I is $\mathbf{t}$ if $I \cap X \in C$ and $\mathbf{f}$ otherwise. Abstract constraints are a generalisation of pseudo-Boolean constraints, cardinality constraints, containment constraints, and much more. A *deterministic* logic program with abstract constraint atoms (Marek et al., 2008) is a set of rules of the form[1]

$$p \leftarrow a_1 \wedge \cdots \wedge a_n \wedge \neg b_1 \wedge \cdots \wedge \neg b_m,$$

where p is an atom and the a_i and b_i are constraint atoms. Having defined the truth value of a constraint atom $C(X)$ in an interpretation I, an immediate consequence operator can be defined in the standard way:

$$T_{\mathcal{P}}(I) = \{p \mid \exists r \in \mathcal{P} : head(r) = p \wedge body(r)^I = \mathbf{t}\}.$$

[1] The approach by Marek et al. (2008) also includes nondeterministic programs. We come back to this issue in Section 3.4.1.

Grounded models of this operator still represent the same intuitions: an interpretation I is grounded if it admits no unfounded sets, or said differently, if it contains no atoms without external support. Thus, I is grounded if it contains no non-empty set U of atoms such that $body(r)^{I[U:\mathbf{f}]}$ for each rule r with $head(r) \in U$.

Example 3.4.8. Let Σ be the alphabet $\{a, b, c, d\}$. For every i, let $C_{\geq i}$ be the cardinality constraint $\{X \subseteq \Sigma \mid |X| \geq i\}$. Consider the following logic program $\mathcal{P}$ over Σ:

$$\left\{ \begin{array}{ll} a. & b \leftarrow C_{\geq 1}(\Sigma). \\ c \leftarrow \neg C_{\geq 4}(\Sigma). & d \leftarrow C_{\geq 4}(\Sigma). \end{array} \right\}$$

Any interpretation in which d holds is *not* grounded since, taking $U = \{d\}$ yields for every I that $C_{\geq 4}(\Sigma)^{I[U:\mathbf{f}]} = \mathbf{f}$ and thus $d \notin T_{\mathcal{P}}(I[U : \mathbf{f}])$. It can easily be verified that $\{a, b, c\}$ is the only grounded model of $\mathcal{P}$.

This example illustrates that even for complex, abstract extensions of logic programs, groundedness is an intuitive property. Groundedness easily extends to these rich formalisms: the lattice always is the space of interpretations, the immediate consequence operator is defined in the standard way and defining grounded models takes only one line given this immediate consequence operator. This is in sharp contrast with more common semantics of logic programming (such as stable and well-founded semantics) which are often hard(er) to extend to richer formalisms, as can be observed by the many different versions of those semantics that exist for logic programs with aggregates (Ferraris, 2005; Son et al., 2006; Pelov et al., 2007; Faber et al., 2011; Gelfond and Zhang, 2014).

Furthermore, groundedness is closely related to one of the most popular semantics for logic programs with aggregates, namely the FLP-stable semantics defined by Faber et al. (2011). Given an interpretation I, Faber, Pfeifer and Leone (2011) defined the reduct of $\mathcal{P}$ with respect to I as the program $\mathcal{P}^I = \{r \mid r \in \mathcal{P} \wedge I \models body(r)\}$. I is an *FLP-stable model* of $\mathcal{P}$ if it is a subset-minimal fixpoint of $T_{\mathcal{P}^I}$. For a large class of programs, this semantics is equivalent with qrounded model semantics, as the following theorem shows.

Theorem 3.4.9. *Let $\mathcal{P}$ be a logic program with abstract constraint atoms. If for each $p \in \Sigma$, there is at most one rule $r \in \mathcal{P}$ with $head(r) = p$, then I is an FLP-stable model of $\mathcal{P}$ if and only if I is a grounded model of $\mathcal{P}$.*

Proof. In this case, for all I and J, $T_{\mathcal{P}^I}(J) = T_{\mathcal{P}}(J) \cap T_{\mathcal{P}}(I)$ since $\mathcal{P}^I$ is obtained from $\mathcal{P}$ by removing all rules with body false in I and there is at most one rule defining each atom. If I is a supported model of $\mathcal{P}$, i.e., a fixpoint of $T_{\mathcal{P}}$, we find that $T_{\mathcal{P}^I}(J) = T_{\mathcal{P}}(J) \cap I$. It is easy to see that FLP-stable models are supported models of $\mathcal{P}$. Assume I is a supported model. In this case I is

FLP-stable if and only if there is no $J \subsetneq I$ with $T_{\mathcal{P}}(J) \cap I = T_{\mathcal{P}^I}(J) \subseteq J$, i.e., if and only if I is strictly grounded. Now, we know from Corollary 3.2.6 that in the context of logic programming, groundedness and strict groundedness are equivalent, which proves our claim. □

3.4.1 Discussion

Unfounded Sets Unfounded sets were first defined by Van Gelder et al. (1991) in their seminal paper introducing the well-founded semantics. Their definition slightly differs from Definition 3.4.1.

Definition 3.4.10 (GRS-Unfounded set)**.** Let $\mathcal{P}$ be a logic program and $\mathcal{I}$ a three-valued interpretation. A set $U \subseteq \Sigma$ is a *GRS-unfounded set* of $\mathcal{I}$ (with respect to $\mathcal{P}$) if for each rule r with $head(r) \in U$, $body(r)^{\mathcal{I}} = \mathbf{f}$ or $body(r)^{\mathcal{I}[U:\mathbf{f}]} = \mathbf{f}$.

The first difference between 2-unfounded sets and GRS-unfounded sets is that GRS-unfounded sets are defined for three-valued interpretations, while we restricted our attention to (two-valued) interpretations. Our definition easily generalises to three-valued interpretations as well.

Definition 3.4.11 (3-Unfounded set)**.** Let $\mathcal{P}$ be a logic program and $\mathcal{I}$ a three-valued interpretation. A set $U \subseteq \Sigma$ is a 3-*unfounded set* of $\mathcal{I}$ (with respect to $\mathcal{P}$) if for each rule r with $head(r) \in U$, $body(r)^{\mathcal{I}[U:\mathbf{f}]} = \mathbf{f}$.

Lemma 3.4.12. *Let $\mathcal{P}$ be a logic program and I an interpretation. A set $U \subseteq \Sigma$ is a 2-unfounded set of I with respect to $\mathcal{P}$ if and only if it is a 3-unfounded set of I with respect to $\mathcal{P}$.*

Proof. Follows immediately from the definitions. □

This definition formalises the same intuitions as Definition 3.4.1: U is a 3-unfounded set if making all atoms in U false in $\mathcal{I}$ results in a state where none of them can be derived. This definition easily translates to algebra as well.

Proposition 3.4.13. *Let $\mathcal{P}$ be a logic program, $\Psi_{\mathcal{P}}$ Fitting's immediate consequence operator and $\mathcal{I}$ a three-valued interpretation. A set $U \subseteq \Sigma$ is a 3-unfounded set if and only if $\Psi_{\mathcal{P}}(\mathcal{I}[U : \mathbf{f}])_2 \cap U = \emptyset$.*

Proof. Recall that Fitting's operator is defined by

$$\Psi_{\mathcal{P}}(\mathcal{I})_1 = \{a \in \Sigma \mid body(r)^{\mathcal{I}} = \mathbf{t} \text{ for some rule } r \in \mathcal{P} \text{ with } head(r) = a\}$$

$$\Psi_{\mathcal{P}}(\mathcal{I})_2 = \{a \in \Sigma \mid body(r)^{\mathcal{I}} \neq \mathbf{f} \text{ for some rule } r \in \mathcal{P} \text{ with } head(r) = a\}$$

The claim now follows immediately from the definition of $\Psi_{\mathcal{P}}(\mathcal{I})_2$. □

The following proposition relates the two notions of unfounded sets.

Proposition 3.4.14. *Let $\mathcal{P}$ be a logic program, $\mathcal{I}$ a three-valued interpretation and $U \subseteq \Sigma$. The following properties hold.*

- *If U is a 3-unfounded set, then U is a GRS-unfounded set.*
- *If $\mathcal{I}[U : \mathbf{f}]$ is more precise than $\mathcal{I}$, then U is a GRS-unfounded set if and only if U is a 3-unfounded set.*

Proof. The first claim follows directly from the definitions.

If $\mathcal{I}$ and U are chosen such that $\mathcal{I}[U : \mathbf{f}]$ is more precise than $\mathcal{I}$, for every formula φ, the condition $\varphi^{\mathcal{I}} = \mathbf{f}$ or $\varphi^{\mathcal{I}[U:\mathbf{f}]} = \mathbf{f}$ is equivalent with $\varphi^{\mathcal{I}[U:\mathbf{f}]} = \mathbf{f}$. Thus we conclude that in this case the notions of 3-unfounded set and GRS-unfounded set are indeed equivalent, which proves the second claim. □

Thus, for a certain class of interpretations, the two notions of unfounded sets coincide. Furthermore, Van Gelder et al. only use unfounded sets to define the well-founded model construction. It follows immediately from Lemma 3.4 by Van Gelder et al. (1991) that every partial interpretation $\mathcal{I}$ in that construction with GRS-unfounded set U satisfies the condition in the second claim in Proposition 3.4.14. This means that 3-unfounded sets and GRS-unfounded sets are equivalent for all interpretations that are relevant in the original work! Essentially, we provided a new formalisation of unfounded sets that coincides with the old definition on all interpretations used in the original work.

Corollary 3.4.6, which states that grounded models of $\mathcal{P}$ are fixpoints of $T_{\mathcal{P}}$ that permit no proper 2-unfounded sets, might sound familiar. Indeed, it has been shown that an interpretation is a *stable model* of a logic program if and only if it is a fixpoint of $T_{\mathcal{P}}$ and it permits no proper GRS-unfounded sets (Lifschitz, 2008). As shown by Leone et al. (1997), also stable models of *disjunctive* logic programs can be characterised as models that permit no proper GRS-unfounded sets.

Groundedness and Nondeterminism In Section 3.4, we restricted our attention to logic programs with abstract constraint atoms in the *bodies* of rules, and we did not allow them in heads of rules. As argued by Marek et al. (2008), allowing them as well in heads gives rise to a *nondeterministic* generalisation of

the immediate consequence operator. A consistent nondeterministic operator maps every point $x \in L$ to a non-empty set $O(x) \subseteq L$. Extending the notion of groundedness to this nondeterministic setting is out of the scope of this dissertation.

3.5 Grounded Fixpoints in Dung's Argumentation Frameworks and Abstract Dialectical Frameworks

Abstract argumentation frameworks (AFs) (Dung, 1995) are simple and abstract systems to deal with contentious information and draw conclusions from it. An AF is a directed graph where the nodes are arguments and the edges encode a notion of attack between arguments. In AFs, we are not interested in the actual content of arguments; this information is abstracted away. In spite of their conceptual simplicity, there exist many different semantics with different properties in terms of characterisation, existence and uniqueness.

Abstract dialectical frameworks (ADFs) (Brewka and Woltran, 2010; Brewka et al., 2013) are a generalisation of AFs in which not only attack, but also support, joint attack and joint support can be expressed.

Recently, Strass (2013) showed that many of the existing semantics of AFs and ADFs can be obtained by direct applications of AFT. In this section we use the aforementioned study to relate grounded fixpoints to AFs and ADFs. We first do so for the case of AFs and afterwards generalise to ADFs.

3.5.1 Abstract Argumentation Frameworks

An *abstract argumentation framework* Θ is a directed graph (A, R) in which the nodes A represent arguments and the edges in R represent attacks between arguments. We say that a *attacks* b if $(a, b) \in R$. A set $S \subseteq A$ *attacks* a if some $s \in S$ attacks a. A set $S \subseteq A$ *defends* a if it attacks all attackers of a. An *interpretation* of an AF $\Theta = (A, R)$ is a subset S of A. The intended meaning of such an interpretation is that all arguments in S are accepted (or believed) and all arguments not in S are rejected. Interpretations are ordered according to the acceptance relation: $S_1 \leq S_2$ iff $S_1 \subseteq S_2$, i.e., if S_2 accepts more arguments than S_1. There exist many different semantics of AFs which each define different sets of acceptable arguments according to different standards or intuitions. The major semantics for argumentation frameworks can be formulated using two

operators: the *characteristic function* F_Θ, which maps an interpretation S to

$$F_\Theta(S) = \{a \in A \mid S \text{ defends } a\}$$

and the operator U_Θ (U stands for unattacked), which maps an interpretation S to

$$U_\Theta(S) = \{a \in A \mid a \text{ is not attacked by } S\}.$$

An interpretation S is *conflict-free* if it is a postfixpoint of U_Θ ($S \leq U_\Theta(S)$), i.e., if no argument in S is attacked by S. The characteristic function is a monotone operator; its least fixpoint is called the *grounded extension* of Θ. The operator U_Θ is an anti-monotone operator; its fixpoints are called *stable extensions* of Θ. Many more semantics, such as *admissible interpretations*, *complete extensions*, *semi-stable extensions*, *stage extensions* and *preferred extensions* can be characterised using the above operators as well (Dung, 1995; Verheij, 1996; Caminada et al., 2012).

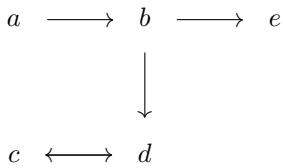
The following proposition shows that grounded extensions as defined in argumentation theory are indeed grounded in the sense defined in this dissertation.

Proposition 3.5.1. *The grounded extension of Θ is the unique grounded fixpoint of F_Θ.*

Proof. Follows immediately from Proposition 3.2.11 which states that a monotone operator has exactly one grounded fixpoint, namely its least fixpoint. □

The grounded extension S consists of all arguments a that should definitely be accepted: all arguments that are globally unattacked, defended by globally unattacked arguments, and so on (recursively). As such, the intuition regarding the grounded extension is similar to intuitions regarding grounded fixpoints: we only accept arguments with a good, non-self-supporting defence.

Example 3.5.2. Consider the following framework:



In this example a is unattacked, hence should be accepted; b is attacked by a, hence should not be accepted. The argument e is defended by a, hence

can safely be accepted. c and d mutually attack each other and hence, defend themselves. Since we have already established that b is rejected, the only remaining argument that defends c is c itself. The grounded extension rejects self-defending arguments (i.e., rejects both c and d) and hence is $\{a, e\}$.

Proposition 3.5.3. *An interpretation S is a stable extension of Θ if and only if it is a (strictly) grounded fixpoint of U_Θ.*

Proof. Follows immediately from Proposition 3.2.12 which states that all postfixpoints of an antimonotone operator are (strictly) grounded. Indeed, U_Θ is anti-monotone and stable extensions are exactly the fixpoints of U_Θ. □

Example 3.5.4 (Example 3.5.2 continued)**.** Stable extensions are more liberal in accepting arguments than the grounded extension. In stable extensions, arguments are "by default" accepted, unless another accepted argument contradicts them. The framework considered in this example has two stable extensions: $\{a, e, c\}$ and $\{a, e, d\}$.

3.5.2 Abstract Dialectical Frameworks

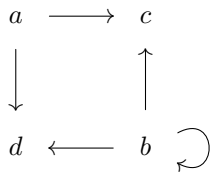
We now extend our theory to the more general case of ADFs. In the context of AFs, grounded fixpoints characterise two existing semantics, when applied to two previously defined operators. In the context of ADFs, however, this does not hold. Here, grounded fixpoints yield a new semantics.

An *abstract dialectical framework* is a triple $\Xi = (S, L, C)$, where

- S is a set of *arguments*
- $L \subseteq S \times S$ is a set of *links*; we define the parents of $s \in S$ as $par(s) = \{r \in S \mid (r, s) \in L\}$,
- $C = \{C_s^{in}\}_{s \in S}$ is a collection of sets C_s^{in} where for every s, $C_s^{in} \subseteq 2^{par(s)}$.

Intuitively, for every s, $par(s)$ is the set of arguments that influence whether or not S should be accepted. This influence can be positive (support), negative (attack) or a combination of both. An argument s should be accepted if for some set $A \in C_s^{in}$, all arguments in A are accepted and all arguments in $par(s) \setminus A$ are not. An argumentation framework $\Theta = (A, R)$ is an ADF in which all links are attack relations, i.e., for every s, $par(s) = \{s' \mid (s', s) \in R\}$ and $C_s^{in} = \{\emptyset\}$ (the only way to accept an argument is if none of its attackers is accepted).

Example 3.5.5. Let S be the set of arguments $\{a, b, c, d\}$ and L the following graph



Furthermore, $C_a^{in} = \{\emptyset\}$, $C_b^{in} = \{\{b\}\}$, $C_c^{in} = \{\emptyset, \{a\}, \{b\}\}$ and $C_d^{in} = \{\{a, b\}\}$. The following observations provide an intuitive reading of the ADF $\Xi = (S, L, C)$.

- a is a valid argument since it has trivial support.
- b supports itself: the only "reason" to believe b is b itself.
- a and b jointly attack c: since $C_c^{in} = \{\emptyset, \{a\}, \{b\}\}$, c is only rejected if a and b are both present
- a and b jointly support d: d is only acceptable if a and b both hold.

Intuitively, we should accept a. Whether or not to accept b, depends on which semantics for ADFs is used. Argument c can only be accepted in case we reject b, d should be accepted if we accept b.

With an ADF Ξ, we associate an operator G_Ξ on the lattice $\langle 2^S, \subseteq \rangle$ as follows (Strass, 2013):

$$G_\Xi(X) = \{s \in S \mid X \cap par(s) \in C_s^{in}\}.$$

This operator generalises the operator U_Θ for AFs.

Strass (2013) showed that many of the existing semantics for ADFs can be characterised with AFT. For example, *models* of Ξ are fixpoints of G_Ξ, *stable models* (Brewka et al., 2013) of Ξ are ultimate stable fixpoints of G_Ξ, et cetera. He also showed that there is a one to one correspondence between ultimate semantics for ADFs and for logic programs, in the sense that every ADF Ξ can be transformed to a logic program $\mathcal{P}$ such that G_Ξ and $T_\mathcal{P}$ coincide and vice versa. It is out of the scope of this dissertation to discuss all of the different semantics for ADFs. Here, we restrict ourselves to discussing the intuitions regarding grounded fixpoints. The intuitions underlying grounded fixpoints of ADFs are of course similar to those in other domains where AFT is applied. Groundedness serves to eliminate "ungrounded" reasoning: if the only reason for accepting an argument is that the argument itself holds, then this argument should be rejected.

Example 3.5.6 (Example 3.5.5 continued)**.** The operator G_Ξ from this example has two fixpoints: $\{a, b, d\}$ and $\{a, c\}$. The first of the two is not a grounded fixpoint because b itself is the only reason to accept b. Formally

$$G_\Xi(\{a, b, d\} \wedge \{a, c, d\}) = G_\Xi(\{a, d\}) = \{a, c\} \leq \{a, c, d\}$$

thus indeed $\{a, b, d\}$ is ungrounded. On the other hand, $\{a, c\}$ is grounded; it is the unique grounded fixpoint of G_Ξ.

We now study groundedness in the context of ADFs.

Definition 3.5.7 (Support)**.** Let $\Xi = (S, L, C)$ be an ADF and $X \subseteq S$. We say that $s \in S$ *has support* in X if $X \cap par(s) \in C_s^{in}$.

Definition 3.5.8 (Grounded model)**.** Let $\Xi = (S, L, C)$ be an ADF and $X \subseteq L$ a model of Ξ. We say that X is a *grounded model* of Ξ if for every set U with $\emptyset \subsetneq U \subseteq X$, at least one $u \in U$ has support in $X \setminus U$.

Thus, a grounded model is one without self-supporting arguments, i.e., without arguments that no longer have support once they are removed.

Below, $\overline{U}$ denotes the set complement of U, i.e., $\overline{U} = S \setminus U$.

Proposition 3.5.9. *Let $\Xi = (S, L, C)$ be an ADF. Then $X \subseteq S$ is a grounded fixpoint of G_Ξ if and only if X is a grounded model of Ξ.*

Proof. The proof is analogous to the proof of Proposition 3.4.5.

First, suppose X is a grounded fixpoint of G_Ξ. Since X is a fixpoint of G_Ξ, by definition it is a model of Ξ. If U is a set $\emptyset \subsetneq U \subseteq X$, we need to show that at least one $u \in U$ has support in $X \setminus U$. Suppose this condition is not satisfied, i.e., that no u has support in $X \setminus U$. This means that $G_\Xi(X \setminus U) \cap U = \emptyset$. Let $V = \overline{U}$; the previous equation translates to $G_\Xi(X \wedge V) \leq V$. Thus, the definition of groundedness yields $X \leq V$, and hence that $U \cap X = \emptyset$, which contradicts with the assumption that $\emptyset \subsetneq U \subseteq X$, hence X is indeed a grounded model of Ξ.

The reverse direction is analogous. Suppose X is a grounded model of Ξ. Let V be such that $G_\Xi(X \wedge V) \leq V$ and let $U = \overline{V} \cap X$. Thus $U \subseteq X$ and (since $X \setminus U = X \wedge V$) $G_\Xi(X \setminus U) \cap U = \emptyset$. Thus, no $u \in U$ has support in $X \setminus U$, hence by the definition of a grounded model, U must be empty, i.e., $X \leq V$ and we conclude that X is indeed a grounded fixpoint in this case. □

It is worth noting that the set U in Definition 3.5.8 corresponds to a proper unfounded set in the case of logic programming.

Many semantics have been defined for ADFs. Most of these semantics are three-valued. The only two-valued semantics are *conflict-free* sets, *supported models* and *two-valued stable models*. Our algebraical results immediately yield that the two-valued stable semantics has the property that it only accepts grounded interpretations. Grounded fixpoints are a new element in the family of two-valued semantics of ADFs. We believe that this is an interesting new member: as illustrated above, it formalises simple and clear intuitions. Two-valued stable semantics and grounded fixpoint semantics formalise related ideas. As with logic programs, we conjecture that for large classes of ADFs these two semantics coincide; it remains an open research question to define those classes (or classes on which they differ). Since Strass (2013) defined transformations between logic programs and ADFs that preserve the operator, solving this research question will also solve the related open question from Section 3.4 and vice versa.

3.5.3 Discussion

Complexity Strass has shown that there is a one to one correspondence between ultimate semantics for ADFs and for logic programs, in the sense that every ADF Ξ can be transformed to a logic program $\mathcal{P}$ such that G_Ξ and $T_\mathcal{P}$ coincide and vice versa. These results allow us to port complexity results from the field of logic programming to ADFs and vice versa. Hence, Theorem 3.4.7 yields that checking existence of a grounded fixpoint of an ADF is Σ_2^P-complete.

In the context of ADFs, one is often also interested in other forms of reasoning such as *credulous* or *sceptical* reasoning (Strass and Wallner, 2014). Analysing complexity of grounded fixpoint semantics for more forms of reasoning is a topic for future work.

3.6 Grounded Fixpoints of Autoepistemic and Default Theories

In this section, we study groundedness in the context of Moore's autoepistemic logic (AEL) (Moore, 1985) and Reiter's default logic (DL) (Reiter, 1980).

In the late seventies, the field of knowledge representation and more particularly, the field of non-monotonic reasoning, increasingly became concerned with the representation of and reasoning on default statements "P's are normally Q's". The idea grew to interpret such statements as defeasible inference rules "if x is a P and it is not known that x is not a Q then (derive that) x is a Q". This idea was developed independently in default logic by Reiter (1980) and nonmonotonic

logic I and II (McDermott and Doyle, 1980; McDermott, 1982). Not much later, Moore (1985) identified the latter sort of statements as *autoepistemic* statements and developed autoepistemic logic (AEL) for it.

In Moore's view, an autoepistemic theory $\mathcal{T}$ is the representation of the knowledge of a perfect, rational, introspective agent. The agent is *introspective* in the sense that propositions in its theory may refer to its own knowledge, through the modal operator K. The informal interpretation of this operator is "I (the agent) know that ...". The agent is a *perfect* reasoner in the sense that its knowledge is closed under entailment. It is *rational* in the sense that it only believes propositions contained in or entailed by its knowledge base $\mathcal{T}$. Thus, $\mathcal{T}$ expresses, directly or indirectly, all the agent knows. Levesque (1990) called this assumption about T the "All I Know Assumption". It is this assumption that distinguishes autoepistemic logic from the standard modal logic of knowledge S5. The challenge in defining such a logic lies in the fact that AEL theories are *self-referential*: what is known by $\mathcal{T}$ is made up from what is expressed by its statements, but what is expressed by a statement depends on what is known by $\mathcal{T}$.

Moore (1985) formalised these ideas as follows. Let $\mathcal{L}$ be the language of propositional logic based on the vocabulary Σ. Extending this language with a modal operator K, yields the language $\mathcal{L}_K$ of modal propositional logic. An *autoepistemic theory* (over Σ) is a set of formulas in $\mathcal{L}_K$. A *modal formula* is a formula of the form $K\psi$, with ψ a formula. An *objective formula* is a formula without modal subformulas.

AEL uses the semantical concepts of standard modal logic. As before, an *interpretation* I is a subset of Σ. It formally represents a potential state of affairs of the world. A *possible world structure* is a set of interpretations. It can be seen as a Kripke structure with the total accessibility relation. The set of all possible world structures, $\mathcal{W}_\Sigma$, is thus $2^{(2^\Sigma)}$. It forms a complete lattice under $\supseteq$. A possible world structure Q formally expresses a potential belief state of an agent by providing all the states of the world that the agent considers to be possible. Interpretations $I \in Q$ are formal representations of possible states of affairs and satisfy the propositions known by the agent. Interpretations $I \notin Q$ represent impossible states of affairs in the sense that they violate some of the agent's propositions.

The semantics of AEL is based on the standard S5 truth assignment. For arbitrary formula φ in $\mathcal{L}_K$, Q a possible world structure and I an interpretation, we define that φ is satisfied with respect to Q and I (denoted $Q, I \models \varphi$) by the standard recursive rules of propositional satisfaction, augmented with one

additional rule:

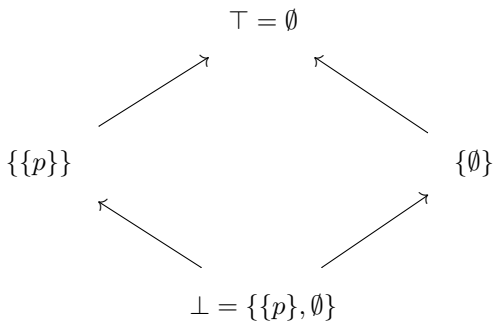
$$Q, I \models K\varphi \text{ if } Q, I' \models \varphi \text{ for every } I' \in Q.$$

For objective formulas φ, it holds that $Q, I \models \varphi$ if and only if $I \models \varphi$. We define $Q \models K\varphi$ (φ is known in Q) if $Q, I \models \varphi$ for every $I \in Q$. As can be seen from the definition of satisfaction, modal formulas are evaluated with respect to the possible world structure Q, while objective formulas are evaluated with respect to the world I.

Example 3.6.1. Consider a formula $\varphi = \neg p \wedge \neg Kp$. Let Q be the possible world structure $\{\{p\}, \emptyset\}$ and let $I = \emptyset$. Then, $Q, I \not\models p$, and $Q, I \not\models Kp$, hence $Q, I \models \varphi$.

The class $\mathcal{W}_\Sigma$ of possible world structures exhibits a natural knowledge order. Intuitively Q contains less knowledge than Q' if it has more possible worlds. Formally we define $Q \leq_k Q'$ if $Q \supseteq Q'$. The intuition underlying this order is clarified by considering the concept of the *objective theory* of a possible world structure Q. This is the set of objective formulas that are known in Q. Formally, it is defined as $Th_{obj}(Q) = \{\varphi \in \mathcal{L} \mid Q \models K\varphi\} = \{\varphi \in \mathcal{L} \mid \forall I \in Q : I \models \varphi\}$. Moore (1984) proved that the function Th_{obj} induces a one-to-one correspondence between possible world structures and sets of objective formulas closed under logical consequence. An obvious property is that $Q \leq_k Q'$ if and only $Th_{obj}(Q) \subseteq Th_{obj}(Q')$. Thus, if $Q \leq_k Q'$ then indeed Q possesses less knowledge than Q'.

With the order $\leq_k$, $\mathcal{W}_\Sigma$ forms a complete lattice. For example, if $\Sigma = \{p\}$, the associated lattice is:



Moore proposed to formalise the intuition that an AEL theory $\mathcal{T}$ expresses "all the agent knows" in semantical terms, as a condition on the possible world

structure Q representing the agent's belief state. The condition is as follows: a world I is possible according to Q if and only if I satisfies $\mathcal{T}$ given Q, that is if $Q, I \models \mathcal{T}$. Equivalently, I is impossible if and only if I violates $\mathcal{T}$ given Q, or $Q, I \not\models \mathcal{T}$. Formally, Moore defines that Q is an *autoepistemic expansion* of $\mathcal{T}$ if for every world I, it holds that $I \in Q$ if and only if $Q, I \models \mathcal{T}$.

The above definition is essentially a fixpoint characterisation. The underlying operator $D_{\mathcal{T}}$ is:

$$D_{\mathcal{T}}(Q) = \{I \mid Q, I \models \mathcal{T}\}.$$

Clearly, Q is an autoepistemic expansion of T if and only if Q is a fixpoint of $D_{\mathcal{T}}$. These autoepistemic expansions are the possible world structures that, according to (Moore, 1985) express candidate belief states of an autoepistemic agent with knowledge base $\mathcal{T}$. Moore called such structures *grounded*.

Soon, researchers such as Halpern and Moses (1985) and Konolige (1988) pointed out certain "anomalies" in the expansion semantics. The simplest example is the theory $\mathcal{T} = \{Kp \Rightarrow p\}$. One of its expansions is $Q_2 = \{\{p\}\}$. The problem with Q_2 is that it is *self-supporting*: Q_2's assumption that p is known to be true, is essential for deriving p. Even from Moore's perspective there might be a problem with such self-supporting belief states. In the first part of his work (Moore, 1985), he argues that sets of inference rules such as the theories that arise in nonmonotonic reasoning, correspond to autoepistemic theories. Viewed from this perspective, $\mathcal{T}$ is the singleton set consisting of one inference rule:

$$\frac{\vdash p}{p}$$

Surely, such an inference rule should be of no value, as it can only derive something that has been derived before! Therefore, the only acceptable belief state for $\mathcal{T}$ seems to be Q_1, the state of total ignorance.

Several attempts were done to strengthen Moore's semantics. Halpern and Moses (1985) proposed an alternative possible world semantics in which the model of an AEL theory $\mathcal{T}$ is the $\leq_k$-least prefixpoint of $D_{\mathcal{T}}$, if it exists. Unfortunately, many simple and natural AEL theories have no model in this semantics. An example is $\{\neg Kp \Leftrightarrow q\}$ for which several prefixpoints of $D_{\mathcal{T}}$ exist but no least one. Here, Moore's semantics makes sense. The unique expansion $\{\{q\}, \{p, q\}\}$ captures the idea that there is no objective information about p, hence p is unknown; therefore, q holds.

Also Konolige (1988) attempted to refine Moore's semantics. He called expansions *weakly grounded* and proposed alternative definitions for so called *moderately grounded* and *strongly grounded* expansions. Intuitively, a possible world structure Q is moderately grounded if all the information it contains can be derived from $\mathcal{T}$ using only ignorance statements from Q. Thus, Q is

moderately grounded if all knowledge in Q follows from $\mathcal{T}$ augmented with all statements of the form $\neg K\varphi$ such that $Q \not\models K\varphi$ (in standard S5 entailment). Konolige proved that moderate grounded expansions are exactly the minimal fixpoints of $D_{\mathcal{T}}$. However, he pointed out that even moderately grounded expansions can give rise to ungrounded reasoning. He illustrated this with the following example.

Example 3.6.2. Consider the following theory

$$\mathcal{T} = \{\neg Kp \Rightarrow q, Kp \Rightarrow p\}.$$

This theory has two moderately grounded possible world structures, namely $Q_1 = \{\{p\}, \{p, q\}\}$ and $Q_2 = \{\{q\}, \{q, p\}\}$. Q_1 is the possible world structure in which p is known ($Kp, \neg K\neg p$) and q is not known ($\neg Kq, \neg K\neg q$).

Again, Konolige argued that Q_1 from Example 3.6.2 should not be grounded. Indeed, in Q_1, the knowledge of p is self-supported. The intended model here is $Q_2 = \{\{q\}, \{q, p\}\}$.

This motivated him to propose the strengthened notion of *strongly grounded* expansion; for a formal definition, see Konolige (1988). The disadvantage of this notion, as recognised by Konolige, is that it is only defined for theories in a normal form where every sentence is of the form

$$K\alpha \wedge \neg K\beta_1 \wedge \cdots \wedge \neg K\beta_n \Rightarrow \gamma,$$

where α, γ and the β_i are objective. Furthermore, whether or not a possible world structure is strongly grounded depends on which transformation to the normal form is used. In other words, strong groundedness is syntactically defined and may hold for one theory and not for an equivalent theory.

Example 3.6.3. Consider theories $\mathcal{T}_1 = \{p\}$ and $\mathcal{T}_2 = \{Kp \Rightarrow p, \neg Kp \Rightarrow p\}$. These theories are equivalent in the modal logic S5. However, $\{\{p\}\}$ is a strongly grounded expansion of $\mathcal{T}_1$, while it is not a strongly grounded expansion of $\mathcal{T}_2$.

We now investigate how the algebraical concept of grounded fixpoint translates to the setting of AEL and how it relates to the above ideas.

Definition 3.6.4 (Unfounded set of impossible worlds)**.** Let $\mathcal{T}$ be an AEL theory and Q a possible world structure. A non-empty set U of worlds is an *unfounded set of impossible worlds* of Q if $U \cap Q = \emptyset$ and for all $I \in U$: $(Q \cup U), I \models \mathcal{T}$.

If Q admits such a U, it contains unsupported knowledge. In particular, the knowledge that the worlds in U are impossible is unsupported, since if we

weaken Q by accepting U as possible worlds, then none of the worlds of U can be dismissed as impossible. All of them satisfy $\mathcal{T}$ in the revised belief state $Q \cup U$. The desired property that an agent's knowledge will satisfy is thus that Q does not admit such a U.

Definition 3.6.5 (Grounded expansion)**.** Let $\mathcal{T}$ be an AEL theory. A possible world structure Q is *grounded for* $\mathcal{T}$ if Q does not admit an unfounded set of impossible worlds. A possible world structure Q is a *grounded expansion* if it is an expansion and grounded.

As it turns out, this notion is again equivalent with the algebraical notion of groundedness.

Proposition 3.6.6. *A possible world structure Q is grounded for $\mathcal{T}$ if and only if Q is (strictly) grounded for $D_{\mathcal{T}}$ in the lattice $\langle \mathcal{W}_\Sigma, \leq_k \rangle$.*

Proof. Definition 3.6.5 can be rephrased by focussing on $Q' = Q \cup U$. Q is grounded if there is no $Q' \supsetneq Q$ such that $Q' \not\subseteq D_{\mathcal{T}}(Q') \cup Q$. We notice that $\supsetneq$ is $<_k$, $\cup$ is $\bigwedge$ and $\subseteq = \geq_k$ in the lattice $\langle \mathcal{W}_\Sigma, \leq_k \rangle$. With this information, Definition 3.6.5 equals the definition of a strictly grounded lattice element. Furthermore, Corollary 3.2.6 guarantees that the notion of strict groundedness coincides with groundedness in powerset lattices. □

The intuitions expressed above correspond closely to those written down by Konolige (1988). On all the examples he gave, groundedness as we defined it, achieves the desired result. Furthermore, since grounded fixpoints are always minimal in $\leq_k$, our notion of groundedness is indeed stronger than the notion of moderate groundedness. We show below (in Corollary 3.6.10) that our notion of groundedness is slightly weaker than strong groundedness. Furthermore, groundedness is defined for every AEL theory (not just for a given normal form) and it is defined purely semantically: two equivalent (in S5) theories have the same grounded expansions.

Example 3.6.7. Consider the following autoepistemic theory:

$$T = \{p, \neg Kp \Rightarrow q, Kq \Rightarrow q\}.$$

The intended possible world structure is clear here: p follows from T using the first sentence, hence p is known. The second sentence cannot be used to derive q, since Kp holds. Furthermore, the last sentence cannot be used to derive q since it first requires Kq. This theory has two autoepistemic expansions, namely $Q_1 = \{\{p\}, \{p, q\}\}$ (which corresponds to knowing p, and not knowing whether q holds or not) and $Q_2 = \{\{p, q\}\}$ (knowing both p and q). The first

one is grounded, while the second is not (it is not even a minimal fixpoint since $Q_1 \leq_k Q_2$). Indeed, if we remove the knowledge that q holds from Q_2 (i.e., we turn the previously impossible world $\{p\}$ into a possible world by adding it to Q_2), then $\{p\}$ remains possible; that is, the belief that $\{p\}$ is impossible is not derived anymore. Hence Q_2 is not grounded.

3.6.1 Groundedness of the AFT family of semantics for AEL

As we saw, the problem of ungrounded expansions remained unsolved for several years. A new take at it was obtained when DMT applied AFT to AEL. We explain this approach.

The bilattice of $\mathcal{W}_\Sigma$ consists of pairs (P, C) of possible worlds. Intuitively, such pairs approximate possible world structures Q such that $C \subseteq Q \subseteq P$, i.e., $P \leq_k Q \leq_k C$: therefore, C is to be understood as a set of *certainly possible worlds* and P as a set of *possibly possible worlds*.

For such pairs, the standard 3- and 4-valued Kleene truth valuation of propositional logic can be extended to a truth function $\varphi^{(P,C),I}$ by adding the following rules for modal formulas:

- $K\varphi^{(P,C),I} = \mathbf{t}$ if for all $I' \in P$, $\varphi^{(P,C),I'} = \mathbf{t}$.
- $K\varphi^{(P,C),I} = \mathbf{f}$ if for some $I' \in C$, $\varphi^{(P,C),I'} = \mathbf{f}$.
- Otherwise, $K\varphi^{(P,C),I} = \mathbf{u}$

That is, φ is known in (P, C) if it holds in all possibly possible worlds, it is not known if it does not hold in at least one certainly possible world. Otherwise, it cannot be determined if φ is known.

This truth valuation induces for each autoepistemic theory $\mathcal{T}$ a bilattice operator $A_\mathcal{T}$ that maps pairs (P, C) to (P', C') where

$$C' = \{I \mid \mathcal{T}^{(P,C),I} = \mathbf{t}\} \text{ and}$$

$$P' = \{I \mid \mathcal{T}^{(P,C),I} \neq \mathbf{f}\}$$

Intuitively, the derived certainly possible worlds are those in which $\mathcal{T}$ evaluates to true, and the derived possibly possible worlds are those in which $\mathcal{T}$ does not evaluate to false.

DMT showed that $A_\mathcal{T}$ is an approximator of $D_\mathcal{T}$. Hence, it induces a class of existing and new semantics for AEL: Moore's expansion semantics (supported

fixpoints), Kripke-Kleene expansion semantics (Denecker et al., 1998) (Kripke-Kleene fixpoints), stable extension semantics (stable fixpoints) and well-founded extension semantics (well-founded fixpoints) (Denecker et al., 2003). The latter two were new semantics induced by AFT. As a corollary of Theorem 3.3.4 and Proposition 3.3.1, we obtain the following analysis of groundedness.

Corollary 3.6.8. *Stable and two-valued well-founded extensions of $\mathcal{T}$ are grounded (in the sense of Definition 3.6.5). If the well-founded extension is two-valued, it is the unique stable extension and the unique grounded expansion. If the Kripke-Kleene expansion is two-valued, it is the well-founded extension, the unique expansion, the unique stable extension and it is grounded.*

For the AEL theory $\{Kp \Rightarrow p\}$, the possible world structure $\{\emptyset, \{p\}\}$ is the well-founded and the unique stable extension. It is also the unique grounded expansion.

An example of an AEL theory with a grounded expansion that is not a stable extension is $\{Kp \Rightarrow p, \neg Kp \Rightarrow p\}$. Its unique grounded expansion is $\{\{p\}\}$ but it has no stable extensions and the well-founded extension is three-valued.

3.6.2 Default logic

Similar to McDermott and Doyle (1980), Reiter (1980) proposed to implement defaults "P's are normally Q's" by their defeasible inference rule "If x is known to be a P and it is consistent to believe that it is a Q, then (infer that) x is a Q". Note that, through the standard duality of modal logic, the second condition is equivalent to "it is not known that x is not a Q". A default logic theory consists of sentences of propositional calculus and default expressions of the form:

$$\frac{\alpha : M\beta_1, \ldots, M\beta_n}{\gamma}$$

where $\alpha, \beta_1, \ldots, \beta_n, \gamma$ are expressions of propositional logic and $n \geq 1$. The informal semantics of such an expression is "if α is known, and it is consistent to believe β_1, ..., and β_n, then γ holds". For this logic, Reiter developed his extension semantics. It soon became clear that Reiter's extension semantics had some exquisite features. For example, consider the following default theory:

$$\left\{ \frac{p : M\mathbf{t}}{p} \right\}$$

It has one extension, namely the theory $Th_{obj}(\{\emptyset, \{p\}\})$. Clearly, in this example default logic avoids the ungrounded model that the related AEL theory $\{Kp \Rightarrow$

$p\}$ has. The sort of ungrounded models that existed for AEL were never discovered in DL. In corollary 3.6.9, we show that Reiter's semantics indeed does not allow ungrounded extensions.

There is an obvious correspondence on the informal level between default expressions and AEL formulas. This connection was explicated by Konolige (1988) who proposed to translate a default theory $\mathcal{T}$ to the AEL theory $Kon(\mathcal{T})$ consisting of AEL formulas:

$$K\alpha \wedge \neg K\neg\beta_1 \wedge \cdots \wedge \neg K\neg\beta_n \Rightarrow \gamma$$

However Kon is not equivalence preserving. Indeed, $\left\{\frac{p:}{p}\right\}$ is a counterexample, as it translates to the AEL theory $\{Kp \Rightarrow p\}$ which under Moore's expansion semantics is not equivalent. In fact, Gottlob (1995) showed that no modular translations exist from DL to AEL (but non-modular transformations exist). For a while, it was believed that AEL and DL were quite different logics. Later, DMT (2003) showed that, similar as for AEL, also with a default theory $\mathcal{T}$ it is possible to associate an approximator $A_{\mathcal{T}}$. This induced again the family of AFT semantics for DL which, just like for AEL, included several existing and some new semantics for DL: weak extensions (Marek and Truszczynski, 1989) (supported fixpoints), Kripke-Kleene extensions (Kripke-Kleene fixpoints), Reiter's extensions (Reiter, 1980) (stable fixpoints) and well-founded extension semantics (Baral and Subrahmanian, 1993) (well-founded fixpoint). Only Kripke-Kleene extensions were a new semantics induced by AFT.

Interestingly, it then appeared that $A_{\mathcal{T}} = A_{Kon(\mathcal{T})}$ for every default logic theory $\mathcal{T}$. Thus, a default theory and Konolige's (modular) translation to AEL have identical approximators. Therefore, they induce the same family of semantics. For example, the extensions of a default theory correspond to the stable extensions of its AEL translation. DMT (2011) argued that the different semantics of AEL and DL induced by AFT correspond to different *dialects* of autoepistemic reasoning. The mismatch found between AEL and DL was due to the fact that Reiter's and Moore's semantics formalised different dialects of autoepistemic reasoning. However, Konolige's translation is correct on a deeper level: it preserves equivalence under every dialect of autoepistemic reasoning!

The above exposition not only tells the story of the link between AEL and DL but also provides the formal material for an analysis of groundedness in the context of DL. Given the embedding of DL into AEL, Definition 3.6.5 also defines groundedness in DL and—using the fact that AFT characterises all main semantics of DL—we obtain the following corollary of Theorem 3.3.4 and Proposition 3.3.1.

Corollary 3.6.9. *Reiter's extensions are grounded. If the well-founded extension of a DL theory is two-valued then it is the unique extension and*

the unique grounded weak extension. If the Kripke-Kleene extension of a DL theory is two-valued then it is grounded (and also the well-founded extension, the unique weak extension, the unique Reiter extension).

An example of a DL theory that has no Reiter extensions but has a grounded weak extension is:

$$\left\{ \frac{p : M\mathbf{t}}{p}, \frac{: M\neg p}{p} \right\},$$

which corresponds to the AEL theory $\{Kp \Rightarrow p, \neg Kp \Rightarrow p\}$. Its unique grounded weak extension is $Th_{obj}(\{\{p\}\})$.

To end the discussion of groundedness in AEL and DL, we return to AEL and the strongly grounded expansions defined by Konolige. He defined them for AEL theories consisting of formulas in the following canonical form:

$$K\alpha \wedge \neg K\neg\beta_1 \wedge \cdots \wedge \neg K\neg\beta_n \Rightarrow \gamma$$

Such formulas are exactly the AEL formulas in the range of Kon. Hence, every AEL theory in this canonical form is $Kon(\mathcal{T})$ for some DL theory $\mathcal{T}$. Konolige showed that the strongly grounded expansions of $Kon(\mathcal{T})$ are exactly the Reiter extensions of $\mathcal{T}$. Combining this result with the previous corollary yields the following.

Corollary 3.6.10. *If Q is a strongly grounded AEL expansion of $\mathcal{T}$, then Q is a grounded expansion of $\mathcal{T}$.*

3.7 Conclusion

The concept of groundedness is widespread in various logic and knowledge representation domains. In this chapter, we formalised this as a property of fixpoints of a lattice operator. In a first step, we analysed grounded fixpoints and their relation to other notions of fixpoints, in particular minimal fixpoints and the different sorts of fixpoints classified in approximation fixpoint theory. The main results here are: given an operator O and an approximator A of O, all A-stable fixpoints are grounded for O and all grounded fixpoints of O are minimal fixpoints approximated by the A-well-founded fixpoint.

We then investigated groundedness in logic programming, abstract argumentation frameworks, autoepistemic logic and default logic. In each of these fields, the concept of groundedness had appeared before, although not always under the same name. We showed links with the notion of unfounded set in logic programming, with groundedness in Dung's argumentation frameworks and

various notions of groundedness in autoepistemic logic and default logic. In each logic, we investigated groundedness of the existing semantics and the new semantics induced by grounded fixpoints. For example, in the context of argumentation frameworks, we discovered that grounded fixpoints recover two existing semantics. In the more general abstract dialectical frameworks, grounded fixpoint yield a new semantics with a clear informal semantics: a fixpoint is grounded if it contains no self-supporting arguments. In autoepistemic logic and default logic, groundedness captures intuitions written down by Konolige.

In summary, the main contributions of the presented work are: we presented a generic formalisation of the concept of groundedness, an analysis of groundedness of existing semantics in a range of logics, and last but not least, the definition of the new semantics for operator based logics induced by grounded fixpoints. In comparison with approximation fixpoint theory, grounded fixpoints are defined directly in terms of the original lattice operator and do not require the invention of an approximator. Grounded fixpoint semantics is compact, intuitive and applicable to all operator based logics. Moreover, it is easily extensible. When adding new language constructs, it suffices to extend the operator for them.

Chapter 4

Partial Grounded Fixpoints

The contents of this chapter will be published in the proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI'15).

4.1 Introduction

In the previous chapter, we extended AFT with the notion of a *grounded* fixpoint. We showed that grounded fixpoints are an intuitive concept, closely related to exact stable fixpoints. In the context of logic programming, grounded fixpoints can be characterised using a generalised notion of unfounded set.

Grounded fixpoints are lattice elements; in this chapter, we generalise them to points in the bilattice, while still maintaining the intuitions, the elegance and desirable properties of groundedness. For the case of logic programming, this generalisation boils down to extending groundedness to partial (or three-valued) interpretations. There are several reasons why it is important to generalise a notion as groundedness to a partial context.

The first, and probably most obvious reason, is that sometimes partial fixpoints are a central topic of study themselves. As an example, in argumentation frameworks and abstract dialectical frameworks, most of the studied semantics are three-valued.

A second reason is that studying concepts in the bilattice (i.e., in a partial context) is more general than studying them only in the original lattice. Many of our results are generalisations of results from the previous section. For instance,

in Corollary 3.3.5, we showed that if the well-founded fixpoint is exact, then it is grounded; in this section, we show that the well-founded fixpoint is *always* grounded (Corollary 4.2.8). Another example is that in this section, we show that all (possibly *partial*) consistent stable fixpoints are grounded (Proposition 4.2.4), a generalisation of Corollary 3.3.3 which states that all *exact* stable fixpoints are grounded.

A third reason is that even in domains where we are only interested in two-valued models, partial models can be useful, e.g., as an analysis tool. For example, in answer set programming (Marek and Truszczyński, 1999) almost all specifications are designed for two-valued stable model semantics. Errors in such specifications may lead to logic programs without stable models; they are hard to debug. Partial stable models (Przymusinski, 1991) can help the debugging process as they can sometimes pinpoint the reason for inconsistency. As a trivial example, let Σ be a vocabulary and p a symbol not in Σ. Augmenting any logic program over Σ with the rule

$$p \leftarrow \neg p$$

yields a logic program without stable models. Debugging tools can use the observation that p is undefined in every partial stable model to identify the "mistake" in the above program, namely the extra rule. The technique of debugging knowledge bases based on partial stable models is used for example in the XNMR system (Castro and Warren, 2001). Partial stable models are not only used for debugging. Real-life databases often contain (local) inconsistencies. The partial stable model semantics for deductive databases is more *robust* than the two-valued stable semantics in this case. It only assigns the value *unknown* to atoms in an inconsistent part of the database, hence it still allows inference on the rest of the database (Seipel et al., 1997; Eiter et al., 1997).

Summarised, the main contributions of this chapter are as follows.

1. We extend the notion of groundedness to points in the bilattice.
2. We study the relationship between partial grounded fixpoints and the other fixpoints studied in AFT. An important result in this context is that we find a new characterisation of the well-founded fixpoint based on groundedness.
3. We study how partial grounded fixpoints depend on the choice of an approximator.
4. We apply our theory to logic programming.
5. We study complexity of both credulous and sceptical reasoning of partial grounded model semantics in logic programming.

4.2 Partial Grounded Fixpoints

We start by giving the central definition of this chapter, namely the notion of A-groundedness. This definition is a direct extension of Definition 3.2.1 to bilattice elements. It should be noted that we do not extend the notion of strictly grounded fixpoints. The most important reason is that this concept does not have an obvious natural extension to the bilattice, or at least, we did not find it yet.

Definition 4.2.1. Let A be an approximator of O. A point $(x, y) \in L^c$ is *A-grounded* if for every $v \in L$ with $A(x \wedge v, y \wedge v)_2 \leq v$, also $y \leq v$.

The definition of this concept is a direct generalisation of groundedness for points $x \in L$. It is based on the same intuitions, but applied in a more general context. We again explain the intuitions under the assumption that the elements of L are sets of "facts" and the $\leq$ relation is the subset relation between such sets. In this case, a point $(x, y) \in L^c$ represents a partial set of "facts": the elements of x are the facts that *certainly* belong to the partial set, the elements in y are those that *possibly* belong to the set. Thus, $y \setminus x$ are the *unknown* elements and the complement of y are those that *do not* belong to the set. In this context, a bilattice point (x, y) is A-grounded if all its possible facts (those in y) are sanctioned by the operator A, in the sense that if we remove some elements from the partial set, then A will make at least one of them possible again (i.e., if we make some elements false, at least one of them should be made *not false*). The above definition captures this idea, by using a set $v \in L$ to remove all elements not in v from (x, y): the point $(x \wedge v, y \wedge v)$ does not contain facts in the complement of v; on all other facts, $(x \wedge v, y \wedge v)$ equals (x, y). In an A-grounded point (x, y) removing elements must result in a state where at least one of them is re-derived to be possible: it must hold that $A(x \wedge v, y \wedge v)_2 \not\leq v$. Stated differently, if the removal of elements from (x, y) is not contradicted by A, i.e., applying A results in a state where these elements are still false, then these elements cannot be part of the grounded point ($y \leq v$).

Example 4.2.2. Let $\mathcal{P}$ be the following logic program

$$\left\{ \begin{array}{l} p \leftarrow p. \\ q \leftarrow \neg r. \end{array} \right\}$$

As always, $\Psi_{\mathcal{P}}$ denotes Fitting's partial immediate consequence operator. Any partial interpretation $\mathcal{I}$ with $p^{\mathcal{I}} \neq \mathbf{f}$ is not $\Psi_{\mathcal{P}}$-grounded. To see this, suppose $\mathcal{I} = (I_1, I_2)$ and let $\mathcal{I}'$ denote $(I_1 \cap \{q, r\}, I_2 \cap \{q, r\})$. Then $p^{\mathcal{I}'} = \mathbf{f}$, hence also $p^{\Psi_{\mathcal{P}}(\mathcal{I}')} = \mathbf{f}$. Thus $\Psi_{\mathcal{P}}(\mathcal{I}')_2 \subseteq \{q, r\}$, while $\mathcal{I}_2 \not\subseteq \{q, r\}$ since we assumed that p is not false in $\mathcal{I}$.

In what follows, we study properties of A-grounded (fix)points, including their relation to other fixpoints studied in AFT. A first observation is that for exact points, i.e., bilattice points of the form (x, x), the choice of the approximator A does not matter and that for those points, A-groundedness coincides with Definition 3.2.1.

Proposition 4.2.3. *If A is a symmetric approximator of O, then x is grounded for O if and only if (x, x) is A-grounded.*

Proof. Trivial. Follows immediately from the fact that $A(x, x)_2 = O(x)$ for all $x \in L$ if A is symmetric. □

A second observation is that all partial A-stable fixpoints are A-grounded. This is a generalisation of Corollary 3.3.3 which states that all exact stable fixpoints are grounded.

Proposition 4.2.4. *All consistent (partial) A-stable fixpoints are A-grounded.*

Proof. Suppose $(x, y) \in L^c$ is an A-stable fixpoint, i.e., $x = \mathrm{lfp}(A(\cdot, y)_1)$ and $y = \mathrm{lfp}(A(x, \cdot)_2)$. Also assume that for some v, $A(x \wedge v, y \wedge v)_2 \leq v$. We show that $y \leq v$. Since A is $\leq_p$-monotone and $y \wedge v \leq y$, it holds that

$$A(x, y \wedge v)_2 \leq A(x, y)_2 = y.$$

Analogously, since A is $\leq_p$-monotone and $x \wedge v \leq x$, it holds that

$$A(x, y \wedge v)_2 \leq A(x \wedge v, y \wedge v)_2 \leq v.$$

Combining these two observations, we find that

$$A(x, y \wedge v)_2 \leq y \wedge v,$$

i.e., that $y \wedge v$ is a prefixpoint of the monotone operator $A(x, \cdot)_2$. Since y is the least prefixpoint of this operator, it must hold that $y \leq y \wedge v$, thus also that $y \leq v$, which we needed to show. □

Example 4.2.5. The converse of Proposition 4.2.4 does not hold. Consider the logic program $\mathcal{P}$

$$\left\{ \; p \leftarrow p \vee \neg p. \; \right\}$$

We claim that every consistent bilattice point is $\Psi_{\mathcal{P}}$-grounded in this case. Indeed, if $(x, y) \in L^c$, then for $v = \{p\}$, $y \leq v$ is trivially true, for $v = \emptyset$, it holds that

$$\Psi_{\mathcal{P}}(x \wedge v, y \wedge v)_2 = \Psi_{\mathcal{P}}(\emptyset, \emptyset)_2 = \{p\} \not\leq \emptyset = v.$$

Hence, all points in the bilattice are $\Psi_{\mathcal{P}}$-grounded. Thus, $(\{p\}, \{p\})$ is a $\Psi_{\mathcal{P}}$-grounded fixpoint which is not $\Psi_{\mathcal{P}}$-stable.

A third observation is that the A-well-founded fixpoint is less precise than any A-grounded fixpoint. This property generalises both the fact that the A-well-founded fixpoint approximates all *exact* grounded fixpoints of O (which we proved in Theorem 3.3.4) and the fact that the A-well-founded fixpoint is less precise than any partial A-stable fixpoint (Theorem 23-2 from Denecker et al. (2000)).

Theorem 4.2.6. *The well-founded fixpoint (u, v) of a symmetric approximator A of O is less precise than any A-grounded fixpoint.*

Before we prove this theorem, we show that the second type of refinements in a well-founded induction eliminates only non-grounded bilattice points. Thus, we show that if (a, b') is an unfoundedness refinement of (a, b), then all bilattice points that are more precise than (a, b), but not more precise than (a, b'), are ungrounded.

Lemma 4.2.7. *Let (a, b) and (a, b') be elements of L^c such that $A(a, b')_2 \leq b' \leq b$. Then for every b'' with $b' \lneq b'' \leq b$, (a, b'') is ungrounded.*

Proof. For every such b'' it holds that $A(a \wedge b', b'' \wedge b')_2 = A(a, b')_2 \leq b'$ while $b'' \not\leq b'$. □

Proof of Theorem 4.2.6. Let $(a_i, b_i)_{i \leq \beta}$ be a well-founded induction of A and let (x, y) be an A-grounded fixpoint. We show by induction that for every $i \leq \beta$, $a_i \leq x$ and $y \leq b_i$.

- The results trivially holds for $i = 0$ since $(a_0, b_0) = (\bot, \top)$.
- It is also clear that the property is preserved in limit ordinals.
- Hence, all we need to show is that the property is preserved by A-refinements. Suppose (a', b') is an A-refinement of (a, b) and that $(a, b) \leq_p (x, y)$. We show that $(a', b') \leq_p (x, y)$. We distinguish two cases. If $(a, b) \leq_p (a', b') \leq_p A(a, b)$, then since A is $\leq_p$-monotone and $(a, b) \leq_p (x, y)$, it holds that $A(a, b) \leq_p A(x, y) = (x, y)$, hence also $(a', b') \leq_p (x, y)$. For the second type of refinements, it follows from Lemma 4.2.7 that only ungrounded bilattice points are removed. □

Te A-well-founded fixpoint is characterised as the least precise A-stable fixpoint. The previous theorem provides us with a similar characterisation of the well-founded fixpoints in terms of groundedness. This is an important result, as it again supports the claim that many of the existing semantics are designed with the intuitions of groundedness in mind: we establish a very tight link between the well-founded fixpoints and grounded fixpoints.

Corollary 4.2.8. *The well-founded fixpoint of a symmetric approximator A of O is the least precise A-grounded fixpoint.*

Proof. The well-founded fixpoint of A is consistent and A-stable, hence Proposition 4.2.4 shows that it is A-grounded as well. Theorem 4.2.6 shows that it is less precise than any A-grounded fixpoint, hence the result follows. □

An operator can have multiple approximators. In Proposition 4.2.3, we already showed that the choice of approximator A does not affect *exact* A-grounded fixpoints: these are exactly the fixpoints that are grounded for O. The question still remains how groundedness of other points in the bilattice is influenced by such a choice. The next proposition answers this question: more precise approximators have fewer grounded points.

Proposition 4.2.9. *If A and B are approximators of O and $A \leq_p B$, then all consistent B-grounded points are also A-grounded.*

Proof. Suppose $(x, y) \in L^c$ is B-grounded; we show that it is also A grounded. Assume that for some v it holds that $A(x \wedge v, y \wedge v)_2 \leq v$. Since $A \leq_p B$, it holds that

$$B(x \wedge v, y \wedge v)_2 \leq A(x \wedge v, y \wedge v)_2 \leq v.$$

Since (x, y) is B-grounded, we find that $y \leq v$. □

Example 4.2.10. The choice of an approximator can really make a difference for the notion of A-groundedness, i.e., not all A-grounded points are B-grounded in Proposition 4.2.9. Consider the logic program $\mathcal{P}$

$$\left\{ \begin{array}{l} p \leftarrow \neg p. \\ q \leftarrow q. \end{array} \right\}$$

Then $(\{q\}, \{p, q\})$ is not $\Psi_{\mathcal{P}}$-grounded since

$$\begin{aligned} \Psi_{\mathcal{P}}(\{q\} \wedge \{p\}, \{p, q\} \wedge \{p\})_2 &= \Psi_{\mathcal{P}}(\emptyset, \{p\})_2 \\ &= (\emptyset, \{p\})_2 \\ &= \{p\}, \end{aligned}$$

while $\{p, q\} \not\leq \{p\}$.

However, let A be the trivial approximator of $T_{\mathcal{P}}$, i.e., the approximator such that $A(x, x) = (T_{\mathcal{P}}(x), T_{\mathcal{P}}(x))$ for every $x \in L$ and $A(x, y) = (\bot, \top)$ for $(x, y) \in L^c$ with $x \neq y$. We claim that $(\{q\}, \{p, q\})$ is A-grounded. We prove this claim by

showing that all $v \in L$ satisfy the condition from Definition 4.2.1. For $v = \top$ the condition $y \leq \top$ is trivially satisfied. For $v = \{q\}$, we find

$$\begin{aligned} A(\{q\} \wedge \{q\}, \{p,q\} \wedge \{q\})_2 &= A(\{q\}, \{q\})_2 \\ &= (\{p,q\}, \{p,q\})_2 \\ &= \{p,q\} \\ &\not\leq \{q\}. \end{aligned}$$

For $v = \{p\}$, we find

$$\begin{aligned} A(\{q\} \wedge \{p\}, \{p,q\} \wedge \{p\})_2 &= A(\emptyset, \{p\})_2 \\ &= (\bot, \top)_2 \\ &= \top \\ &\not\leq \{p\} \end{aligned}$$

For $v = \emptyset$, we find that

$$\begin{aligned} A(\{q\} \wedge \emptyset, \{p,q\} \wedge \emptyset)_2 &= A(\emptyset, \emptyset)_2 \\ &= (\{p\}, \{p\})_2 \\ &= \{p\} \\ &\not\leq \emptyset. \end{aligned}$$

This indeed proves our claim.

4.3 Partial Grounded Fixpoints in Logic Programming

In this section, we apply our theory to logic programming. More concretely we extend the observation from Section 3.4 that in logic programming, groundedness is closely related to unfounded sets. Here, we define several variants of unfounded sets (one for each approximator of $T_{\mathcal{P}}$) and show that A-groundedness is directly characterised by means of A-unfounded sets. When taking for A Fitting's (partial) immediate consequence operator, our definition coincides with the notion of "3-unfounded set" from Definition 3.4.11.

Intuitively, an unfounded set is a set of atoms that might circularly support themselves, but have no support from outside. Stated differently, an unfounded set of a logic program $\mathcal{P}$ with respect to a partial interpretation $\mathcal{I}$ is a set U of atoms such that $\mathcal{P}$ provides no support for any atom in U if the atoms in U are assumed false. Each approximator A of the immediate consequence operator $T_\mathcal{P}$ defines its own notion of support: it maps a partial interpretation $\mathcal{I}$ to $A(\mathcal{I})$, where $A(\mathcal{I})_1$ are the atoms that are supported by $\mathcal{P}$, and $A(\mathcal{I})_2$ are the atoms that are possibly supported by $\mathcal{P}$, depending on the values of atoms unknown in $\mathcal{I}$. Thus, the atoms for which A provides no support are those not in $A(\mathcal{I})_2$ and the above intuitions are formalised directly as follows.

Definition 4.3.1 (A-Unfounded set)**.** Let $\mathcal{P}$ be a logic program, A an approximator of $T_\mathcal{P}$ and $\mathcal{I}$ a three-valued interpretation. A set $U \subseteq \Sigma$ is an *A-unfounded set* of $\mathcal{P}$ with respect to $\mathcal{I}$ if $A(\mathcal{I}[U : \mathbf{f}])_2 \cap U = \emptyset$.

When A is Fitting's immediate consequence operator, our definition coincides with the notion of 3-unfounded set from Definition 3.4.11. We already showed that for partial interpretations $\mathcal{I}$ such that $\mathcal{I}[U : \mathbf{f}]$ is more precise than $\mathcal{I}$, $\Psi_\mathcal{P}$-unfounded sets are exactly unfounded sets as defined by Van Gelder et al. (1991) and that all interpretations in the well-founded model construction satisfy this condition.

Example 4.3.2 (Example 4.2.2 continued)**.** Let $\mathcal{P}$ again be

$$\left\{ \begin{array}{l} p \leftarrow p. \\ q \leftarrow \neg r. \end{array} \right\}$$

In this case, $\{p\}$ is a $\Psi_\mathcal{P}$-unfounded set of any partial interpretation $\mathcal{I}$. Indeed, as argued in Example 4.2.2, p is false in $\Psi_\mathcal{P}(\mathcal{I}[\{p\} : \mathbf{f}])$.

We now show how groundedness relates to this generalised notion of unfounded sets. This proposition generalises Proposition 3.4.5.

Proposition 4.3.3. *Let $\mathcal{P}$ be a logic program, and A an approximator of the immediate consequence operator $T_\mathcal{P}$. A partial interpretation $\mathcal{I}$ is A-grounded if and only if all atoms that belong to an A-unfounded set U of P with respect to $\mathcal{I}$ are false in $\mathcal{I}$.*

Proof. First, suppose $\mathcal{I} = (I_1, I_2)$ is A-grounded and U is an A-unfounded set of $\mathcal{P}$ with respect to $\mathcal{I}$. Let $V = \Sigma \setminus U$ be the complement of U. Since U is an A-unfounded set, $A(\mathcal{I}[U : \mathbf{f}])_2 \cap U = \emptyset$. This means that $A(I_1 \wedge V, I_2 \wedge V)_2 \leq V$, hence the definition of A-groundedness yields $I_2 \leq V$: all elements of U are false in I.

The reverse direction is analogous. Suppose every A-unfounded set is disjoint from I_2. Let V be such that $A(I_1 \wedge V, I_2 \wedge V)_2 \leq V$ and let $U = \Sigma \setminus V$ be the complement of V. Then again $\mathcal{I}[U : \mathbf{f}] = (I_1 \wedge V, I_2 \wedge V)$ hence U is an A-unfounded set of P with respect to $\mathcal{I}$. Thus, it must hold that $I_2 \cap U = \emptyset$, i.e., that $I_2 \leq V$. We conclude that $\mathcal{I}$ is indeed A-grounded. □

Following the analogy with other partial semantics of logic programs, we call a partial interpretation $\mathcal{I}$ such that $\mathcal{I}$ is a $\Psi_\mathcal{P}$-grounded fixpoint of $T_\mathcal{P}$ a *partial grounded model* of $\mathcal{P}$. We briefly discuss complexity of partial grounded model semantics. First of all, deciding whether $\mathcal{P}$ has a partial grounded model is not an interesting task since the well-founded model is always grounded. We study other deterministic inference tasks, namely *credulous* and *sceptical* query answering (sometimes also called *possibility inference* and *certainty inference*, respectively) (Schlipf, 1995; Abiteboul et al., 1990; Saccà, 1997) under the (partial) grounded semantics. The first of these tasks consists of deciding whether a symbol $p \in \Sigma$ holds in *some* grounded model, the second consists of deciding whether it holds in *all* grounded models.

Theorem 4.3.4. *Given a finite propositional logic program $\mathcal{P}$ over Σ and an atom $p \in \Sigma$, the following hold.*

1. *The problem of deciding whether p holds in* some *partial grounded model of $\mathcal{P}$ is Σ_2^P-complete.*
2. *The problem of deciding whether p holds in* all *partial grounded models of $\mathcal{P}$ is in P.*

The proof of the first point is analogous to the proof of Theorem 3.4.7.

Proof. This first problem is in Σ_2^P; this follows from the fact that verifying whether a partial interpretation $\mathcal{I}$ is a partial grounded model can be done by calculating $\Psi_\mathcal{P}(\mathcal{I}[U : \mathbf{f}])$ for all candidate $\Psi_\mathcal{P}$-unfounded sets.

We now show Σ_2^P-hardness of the problem of existence of a (partial) grounded model of a program $\mathcal{P}$ in which p holds. Let φ be propositional formula in DNF over propositional symbols $x_1, \ldots, x_m, y_1, \ldots, y_n$. For an interpretation $I \subseteq \{x_1, \ldots, x_m\}$, we define φ_I as the formula obtained from φ by replacing all atoms $x_i \in I$ by $\mathbf{t}$ and all atoms $x_i \notin I$ by $\mathbf{f}$. Recall that the problem of deciding whether there exists an interpretation $I \subseteq \{x_1, \ldots, x_m\}$ such that φ_I is a tautology is Σ_2^P-hard. We now reduce this problem to our problem. For each x_i, we introduce a new variable x_i' with as intended meaning the negation of x_i. Let φ' be the formula obtained from φ by replacing all literals $\neg x_i$ by

x'_i. In the proof of Theorem 3.4.7, we defined a program $\mathcal{P}(\varphi)$ consisting of the following clauses

1. $x_i \leftarrow \neg x'_i$ and $x'_i \leftarrow \neg x_i$ for each $i \in \{1, \ldots, m\}$,
2. $y_i \leftarrow \varphi'$ for each $i \in \{1, \ldots, n\}$,
3. $p \leftarrow \varphi'$,
4. $q \leftarrow \neg p \wedge \neg q$.

We need to show that there is an $I \subseteq \{x_1, \ldots, x_m\}$ such that φ_I is a tautology if and only if $P(\varphi)$ has a partial grounded model in which p holds.

In the proof of Theorem 3.4.7, we already showed that there is an $I \subseteq \{x_1, \ldots, x_m\}$ such that φ_I is a tautology if and only if $P(\varphi)$ has an exact grounded model. Hence, all we need to show is that if $P(\varphi)$ has a partial stable fixpoint, then there is such an I as well. We will prove a stronger claim, namely that every partial grounded model of $\mathcal{P}(\varphi)$ in which p holds can be extended to a two-valued grounded model in which p holds.

It is easy to see that in each partial fixpoint M of $\Psi_{\mathcal{P}(\varphi)}$ with $p^M = \mathbf{t}$ the following properties hold:

1. $y_1, \ldots, y_n$ are true in M (since their rules have the same body as p),
2. for each $i \in \{1, \ldots, m\}$, either both x_i and x'_i are unknown in M or one of them is true and the other is false in M.

Now, suppose M is a partial grounded model of $\mathcal{P}$. By the above observations, the only unknown atoms in M can be q, the x_i and the x'_i. Now, it is easy to see that $M[\{q\} : \mathbf{f}]$ is also a grounded model of $\mathcal{P}$. Similarly, if x_i and x'_i are unknown in M, then $M[\{x_i\} : \mathbf{t}, \{x'_i\} : \mathbf{f}]$ is a grounded model as well. Hence, our claim follows.

The second point follows from the fact that the well-founded model of $\mathcal{P}$ can be computed in polynomial time (Van Gelder et al., 1991) and that the well-founded model is the least precise partial grounded model (Corollary 4.2.8). □

4.4 Discussion

In the context of logic programming and argumentation frameworks, many subclasses of partial stable models have been proposed.

- A partial stable model (X, Y) of $\mathcal{P}$ is called *M-stable* (for maximally stable) if it is $\leq_p$ maximal, i.e., if there is no more precise partial stable model (Saccà and Zaniolo, 1997). These models coincide with the preferred extensions of Dung (1995) and the regular models of You and Yuan (1990) for non-disjunctive logic programs.
- A partial stable model (X, Y) of $\mathcal{P}$ is called *L-stable* (for least undefined stable) if $Y \setminus X$ is $\subseteq$-minimal, i.e., if there exists no partial stable model (X', Y') of $\mathcal{P}$ such that $Y' \setminus X' \subsetneq Y \setminus X$ (Saccà and Zaniolo, 1997). L-stable models only differ from exact stable models if a program has no two-valued stable models. This property is particularly useful in the context of debugging.

Recent work by Strass (2013) lifted these two notions to approximation fixpoint theory.[1] His definitions can be used immediately to define partial M-grounded and L-grounded fixpoints as well. Studying their properties, e.g., complexity, is a topic for future work.

4.5 Conclusion

In this chapter, we extended groundedness to points in the bilattice: for every approximating operator A, we defined a notion of A-groundedness. We showed that for exact lattice points, this coincides with groundedness for O. We related A-grounded fixpoints to other fixpoints in AFT: all A-stable fixpoints are A-grounded and the A-well-founded fixpoint approximates all A-grounded points. This gave us a new characterisation of the A well-founded fixpoint: it is the least precise A-grounded fixpoint.

We applied our algebraical theory to logic programming, where A-groundedness is closely related to unfounded sets.

[1] In order to define L-stable fixpoints, some additional assumptions about the lattice need to be made.

Chapter 5

On Well-Founded Set-Inductions and Locally Monotone Operators

This chapter contains new, unpublished, work.

5.1 Introduction

The informal and formal semantics of nonmonotonic logics such as logic programming and autoepistemic logic have been a topic of at moments intense debate for several decades. Many semantics have been defined for these logics. In several foundational papers, Denecker and coauthors argued for a constructive interpretation of these logics and that well-founded inductions provide the desired construction. They went as far as to suggest that for 'sound' theories, the well-founded construction always is strong enough to construct the intended (exact) model, and that if the well-founded construction gets trapped in a non-exact state, this is sign of a semantical anomaly in the theory which should be considered to be flawed. A well-known example of a "flawed" program is the following

$$\{p \leftarrow \neg p.\}.$$

This claim was made and proven for logic programs viewed as inductive definitions (Denecker and Vennekens, 2014). Such a claim was also made for autoepistemic logic (Denecker et al., 2011). However, no proof was given

for that case. Shortly after the latter paper was published, it became clear that for autoepistemic logic, this claim was incorrect: the well-founded model construction is not nearly strong enough to always construct the intended model.

Example 5.1.1. In 2011, Hanne Vlaeminck, then PhD student of Denecker, presented the following, so far unpublished example to show the weakness of the well-founded semantics of autoepistemic logic.

Consider the autoepistemic theory

$$\{q \Leftrightarrow \neg Kp, r \Leftrightarrow \neg Kq\}.$$

Since p does not occur objectively in $\mathcal{T}$, an agent who only knows $\mathcal{T}$ does not have any information about p. Thus $\neg Kp$ and $\neg K\neg p$ must hold in the intended model. The first sentence then entails q, hence Kq must hold. Now, the last sentence implies $\neg r$; the intended model is thus $\{\{p, q\}, \{q\}\}$, the unique possible world structure in which $\neg Kp, \neg K\neg p, Kq$, and $K\neg r$ hold.

Hanne discovered this example when investigating a translation from ordered epistemic logic (OEL) to AEL. The above AEL theory was obtained by applying this translation to some OEL theory with the intended interpretation $\{\{p, q\}, \{q\}\}$. Thus, this example showed not only the translation was not equivalence preserving, but also the weakness of the well-founded semantics, something that her (and some of my) supervisors—who had just advocated the well-founded semantics—certainly had not expected.

We make some observations regarding Example 5.1.1.

1. As we show in Section 5.3, the well-founded semantics fails to identify the intended model.
2. We can, informally, construct the intended model following some sort of stratification (first reasoning on knowledge of p, next on knowledge of q and finally on knowledge of r).
3. $D_{\mathcal{T}}$ has a unique fixpoint, which is the intended model and which is grounded.

DMT (2011) have strongly argued in favour of a constructive semantics for AEL. While a constructive semantics has indeed important advantages, the above example shows that the constructive semantics of their choice, the well-founded semantics, is too weak for making the construction. An alternative way suggested by the last observation is to use the grounded fixpoint semantics for AEL instead. However, the grounded fixpoint semantics is non-constructive.

In order to solve this discrepancy, in this chapter we define a novel and stronger constructive semantics for AEL. We show that on a large class of theories (*monotonically stratified* theories, a class that generalises Example 5.1.1), our stronger construction is capable to construct a unique model. Moreover, we will prove that this model is the unique grounded fixpoint of the semantic operator $D_{\mathcal{T}}$. This gives strong evidence that indeed, the presented construction fits better with the aims of DMT.

In Section 5.4, we define—algebraically—a refinement of the well-founded semantics. Our refined semantics is constructive and differs from the well-founded semantics in the sense that instead of approximating lattice elements by intervals (bilattice elements), we use arbitrary sets of lattice elements. In Section 5.5, we introduce a class of lattice operators, called *locally monotone* operators. We prove that for locally monotone operators, our refined semantics yields a single fixpoint, the unique grounded fixpoint of the operator. In Section 5.6, we show that monotonically stratified AEL theories induce a locally monotone operator and that the unique fixpoint identified by our semantics coincides with the intended model for monotonically stratified theories. Thus, we show that the grounded fixpoint semantics is correct for monotonically stratified theories and provide a general, algebraical, construction of the unique grounded fixpoint.

In Section 5.7, we briefly apply our theory to logic programming. In this context, we show that (locally) stratified logic programs (Przymusinski, 1988) induce a locally monotone operator.

In Section 5.8, we relate our theory to the work by Vennekens et al. (2006) on modularity of lattice operators and the work by Niemelä (1991) on a constructive semantics for AEL. We conclude in Section 5.9.

In this chapter, we restrict our attention (e.g., in Definition 5.5.27) to finite stratifications since the infinite case (n is an ordinal) requires several technical details without facilitating new insights. These technical details can be found in Appendix A. In the appendix, we show that our main technical results, namely Theorems 5.5.28 and 5.5.29 also hold in the infinite case.

5.2 Preliminaries

If φ is a AEL-formula, $At(\varphi)$ denotes the set of all atoms that occur in φ and $At_O(\varphi)$ the set of all atoms that occur objectively, i.e., not in the scope of an operator K, in φ.

Recall from Section 3.6 that the lattice of all possible world structures is denoted $\mathcal{W}_\Sigma$ and that the semantic operator $D_{\mathcal{T}}$ of an autoepistemic theory is defined

by

$$D_{\mathcal{T}}(Q) = \{I \mid Q, I \models \mathcal{T}\}.$$

Following DMT (2003), the semantics of an autoepistemic theory is defined by an approximator on the bilattice $\mathcal{W}^2_\Sigma$. In this chapter, we use the ultimate approximator $U_{\mathcal{T}}$ since this is the most precise approximator and we intend to show that even the ultimate well-founded model is not precise enough to capture the intended semantics of a class of autoepistemic theories. If we show this for the ultimate approximator, the same also trivially follows for all other approximators of $D_{\mathcal{T}}$. $U_{\mathcal{T}}$ maps (P, S) to (P', S'), where

$$P' = \{I \mid Q, I \models \mathcal{T} \text{ for a } Q \text{ with } P \supseteq Q \supseteq S\}$$

$$S' = \{I \mid Q, I \models \mathcal{T} \text{ for all } Q \text{ with } P \supseteq Q \supseteq S\}$$

DMT (2011) gave compelling arguments defending the (ultimate) well-founded semantics for autoepistemic reasoning. Well-founded inductions mimic the reasoning process of a rational agent: an (ultimate) well-founded induction $(P_i, Q_i)_{i \leq \beta}$ of $D_{\mathcal{T}}$ gradually derives what the agent knows (statements φ such that $P_i \models K\varphi$) and what it does not know (φ such that $Q_i \not\models K\varphi$).

Following Vennekens et al. (2006), we call an autoepistemic theory $\mathcal{T}$ *stratifiable*[1] with respect to a partition $(\Sigma_i)_{0 \leq i \leq n}$ of its alphabet if there exists a partition $(\mathcal{T}_i)_{0 \leq i \leq n}$ of $\mathcal{T}$ such that for each i, $At_O(\mathcal{T}_i) \subseteq \Sigma_i$ and $At(\mathcal{T}_i) \subseteq \bigcup_{0 \leq j \leq i} \Sigma_j$. This notion of stratification significantly extends the notion of Marek and Truszczyński (1991). A stratification is *modally separated* if for every modal subformula $K\psi$ of $\mathcal{T}_i$, either $At(\psi) \subseteq \Sigma_i$ or $At(\psi) \subseteq \bigcup_{0 \leq j < i} \Sigma_j$.

Let Σ_1 and Σ_2 be two disjoint vocabularies. If Q_1 and Q_2 are possible world structures over Σ_1 and Σ_2 respectively, then the extension of Q_1 with Q_2 is the possible world structure over $\Sigma_1 \cup \Sigma_2$ defined as $Q_1 \oplus Q_2 \stackrel{\text{def}}{=} \{I_1 \cup I_2 \mid I_1 \in Q_1 \wedge I_2 \in Q_2\}$. If Q is a possible world structure over $\Sigma_1 \cup \Sigma_2$, the restriction of Q to Σ_1 is $Q|_{\Sigma_1} \stackrel{\text{def}}{=} \{I \cap \Sigma_1 \mid I \in Q\}$.

5.3 Motivation

In this section, we first show that for certain AEL theories such Example 5.1.1, the well-founded fixpoint is not precise enough. Afterwards, we generalise this example to obtain a class (called monotonically stratified theories) of theories for which we can unambiguously define the intended model. By analogy with a

[1] As mentioned in the introduction, we restrict ourselves to finite stratifications.

similar concept in logic programming, namely local stratification (Przymusinski, 1988), we call the intended model of a monotonically stratified theory its *perfect model*.

Example 5.3.1 (Example 5.3.1 continued)**.** Again, let $\mathcal{T}$ be the autoepistemic theory

$$\{q \Leftrightarrow \neg Kp, r \Leftrightarrow \neg Kq\}.$$

We now show that the well-founded semantics fails to identify the intended model $\{\{p,q\},\{q\}\}$. Let $U_{\mathcal{T}}$ be the ultimate approximator of $D_{\mathcal{T}}$. Recall that $\top = \emptyset$ and $\bot$ is the possible world structure containing all interpretations. First, we notice that

$$\begin{aligned} U_{\mathcal{T}}(\bot,\top)_1 &= \bigwedge D_{\mathcal{T}}([\bot,\top]) \\ &\leq D_{\mathcal{T}}(\bot) \wedge D_{\mathcal{T}}(\{\{p,q\},\{q\}\}) \wedge D_{\mathcal{T}}(\{\{p\},\{q,p\}\}) \wedge D_{\mathcal{T}}(\top) \\ &= \{\{q,r\},\{q,r,p\}\} \wedge \{\{p,q\},\{q\}\} \wedge \{\{p,r\},\{r\}\} \wedge \{\emptyset,\{p\}\} \\ &= \bot \end{aligned}$$

and that

$$\begin{aligned} U_{\mathcal{T}}(\bot,\top)_2 &= \bigvee D_{\mathcal{T}}([\bot,\top]) \\ &\geq D_{\mathcal{T}}(\bot) \vee D_{\mathcal{T}}(\top) \\ &= \{\{q,r\},\{q,r,p\}\} \vee \{\emptyset,\{p\}\} \\ &= \top. \end{aligned}$$

Hence $U_{\mathcal{T}}(\bot,\top) = (\bot,\top)$ and there are no strict refinements of the first kind of $(\bot,\top)$. Suppose that for some y', $U_{\mathcal{T}}(\bot,y')_2 \leq y'$. Since $D_{\mathcal{T}}(\bot) = \{\{q,r\},\{q,r,p\}\}$, it follows that $y' \geq \{\{q,r\},\{q,r,p\}\}$. We find that

$$\begin{aligned} y' &\geq U_{\mathcal{T}}(\bot,y')_2 \\ &= \bigvee\{D_{\mathcal{T}}([\bot,y'])\} \\ &\geq D_{\mathcal{T}}(\bot) \vee D_{\mathcal{T}}(\{\{q,r\},\{q,r,p\}\}) \\ &\geq \{\{q,r\},\{q,r,p\}\} \vee \{\{q\},\{q,p\}\} \\ &= \emptyset \\ &= \top. \end{aligned}$$

Thus, there are also no strict refinements of the second kind either; $(\bot, \top)$ is the ultimate well-founded fixpoint of $D_{\mathcal{T}}$.

We observe that the theory from the above example is stratifiable with respect to the partition $\langle\{p\}, \{q\}, \{r\}\rangle$. Symbols in a stratum are uniquely defined in terms of knowledge of symbols in lower strata, hence we expect that there is a unique two-valued model which can be constructed *from the ground up*, following the stratification. This is not the case under the (ultimate) well-founded semantics. We now generalise this example.

Definition 5.3.2 (Monotonically stratified)**.** We say that $\mathcal{T}$ is *monotonically stratified* with respect to a partition $(\Sigma_i)_{0\leq i\leq n}$ of its alphabet if there is a modally separated stratification $(\mathcal{T}_i)_{0\leq i\leq n}$ of $\mathcal{T}$ such that all modal subformulas $K\psi$ of $\mathcal{T}_i$ with $At(\psi) \subseteq \Sigma_i$ occur negatively (in the scope of an odd number of negations) in $\mathcal{T}_i$.

The stratification is *strict* if there are no modal subformulas $K\psi$ of $\mathcal{T}_i$ with $At(\psi) \subseteq \Sigma_i$.

The intuition regarding this definition is that whether or not symbols in Σ_i are known by the agent is completely determined by knowledge of symbols in $\Sigma_{i'}$ with $i' < i$ and by $\mathcal{T}_i$. Hence, knowledge of the agent can be built up in strata. Restricted to $\mathcal{T}_i$, symbols in Σ_i only depend positively on their own knowledge. This guarantees that $D_{\mathcal{T}}$ induces a monotonic operator when restricted to Σ_i.

Proposition 5.3.3. *Let $(\mathcal{T}_i)_{0\leq i\leq n}$ be a monotonic stratification of $\mathcal{T}$ with respect to $(\Sigma_i)_{0\leq i\leq n}$. For some i, let Q_{i-1} be a possible world structure over $\bigcup_{j<i} \Sigma_j$. The operator*

$$D_i : \mathcal{W}_{\Sigma_i} \to \mathcal{W}_{\Sigma_i} : Q \mapsto D_{\mathcal{T}_i}(Q \oplus Q_{i-1})|_{\Sigma_i}$$

is monotone.

Proof. Follows from the fact that all modal subformulas $K\psi$ of $\mathcal{T}_i$ with $At(\psi) \subseteq \bigcup_{0\leq j<i} \Sigma_j$ are evaluated with respect to Q_{i-1}, which is fixed, and all other modal formulas occur negatively, i.e., more knowledge can only yield more derivations. □

Proposition 5.3.3 can be used to iteratively build up a possible world structure similar to the perfect model for logic programs (Przymusinski, 1988). Given this similarity, we also call this constructed model *perfect*. Contrary to the case for logic programming, the well-founded model does not always equal the perfect model here, as Example 5.3.1 shows.

Definition 5.3.4 (Perfect model)**.** Let $\mathcal{T}$ be a monotonically stratified autoepistemic theory and $(\mathcal{T}_i)_{0\leq i\leq n}$ a monotonic stratification of $\mathcal{T}$. The *perfect model* of $\mathcal{T}$ (denoted $pm(\mathcal{T})$) is defined by induction on n.

- If $n = 0$, then $D_{\mathcal{T}}$ is monotone and the perfect model of $\mathcal{T}$ is the least fixpoint of $D_{\mathcal{T}}$.
- Otherwise, let Q_{n-1} denote $pm(\bigcup_{j<n} \mathcal{T}_j)$ and let D_n be as in Proposition 5.3.3; in this case we define $pm(\mathcal{T})$ as $\mathrm{lfp}(D_n) \oplus Q_{n-1}$.

Example 5.3.5 (Example 5.3.1 continued)**.** Again, let $\mathcal{T}$ be the autoepistemic theory

$$\{q \Leftrightarrow \neg Kp, r \Leftrightarrow \neg Kq\}.$$

This theory is monotonically stratified with respect the the partition $\langle\{p\}, \{q\}, \{r\}\rangle$. The perfect model of $\mathcal{T}$ is computed as follows.

$$\begin{aligned} pm(\mathcal{T}_0) &= \{\emptyset, \{p\}\} \\ pm(\mathcal{T}_0 \cup \mathcal{T}_1) &= \{\{q\}\} \oplus pm(\mathcal{T}_0) = \{\{q\}, \{q, p\}\} \\ pm(\mathcal{T}_0 \cup \mathcal{T}_1 \cup \mathcal{T}_2) &= \{\emptyset\} \oplus pm(\mathcal{T}_1) = \{\{q\}, \{q, p\}\} \end{aligned}$$

where

$$\begin{aligned} \mathcal{T}_0 &= \{\} \\ \mathcal{T}_1 &= \{q \Leftrightarrow \neg Kp\} \\ \mathcal{T}_2 &= \{r \Leftrightarrow \neg Kq\} \end{aligned}$$

Hence, in this example, the perfect model indeed corresponds to the intended model. Furthermore, the perfect model computation follows the reasoning process described in Example 5.1.1. Furthermore, it can be seen that $\{\{q\}, \{q, p\}\}$ is the unique grounded fixpoint of $D_{\mathcal{T}}$.

We show in Corollary 5.6.5 that the perfect model of every "good" theory is the unique grounded fixpoint of its semantic operator.

The idea underlying the construction of the perfect model is that the knowledge of symbols in Σ_i remains fixed after computing the perfect model of $\bigcup_{j<i} \mathcal{T}_j$. Even though this seems intuitively clear, it is not always the case, as the following example shows.

Example 5.3.6. Let $\mathcal{T}$ be the AEL theory

$$\{p, Kp \Rightarrow q, Kp \Rightarrow \neg q\}.$$

This theory is monotonically stratified with respect to the partition $\langle\{p\},\{q\}\rangle$. The lowest stratum determines the knowledge of p; it consists of the theory $\{p\}$, hence we conclude that the intended model Q of $\mathcal{T}$ satisfies $Q \models Kp$ and $Q \not\models K\neg p$. However, in the second stratum, an inconsistency occurs, since both q and $\neg q$ are derived. Hence the perfect model of $\mathcal{T}$ is the inconsistent possible world structure $\top = \emptyset$. Thus, in the perfect model of $\mathcal{T}$, also $\neg p$ is known, which violates our previous conclusion.

Luckily enough, this strange behaviour only occurs when the inconsistent possible world structure is involved. This kind of peculiarities has already been noted by Vennekens et al. (2007). They introduced the syntactical notion of a *permaconsistent* theory in order to avoid problems with the inconsistent possible world structure. We use a slightly weaker variant.

Definition 5.3.7 (Weakly permaconsistent)**.** An autoepistemic theory $\mathcal{T}$ is called *weakly permaconsistent* if for every possible world structure Q, at least one I satisfies $Q, I \models \mathcal{T}$.

Proposition 5.3.8. *Suppose $\mathcal{T}$ is weakly permaconsistent and $(\mathcal{T}_i)_{0\leq i\leq n}$ is a monotonic stratification of $\mathcal{T}$ with respect to the partition $(\Sigma_i)_{0\leq i\leq n}$. Then, for every objective formula φ over $\bigcup_{j\leq i}\Sigma_j$, $pm(\mathcal{T}) \models K\varphi$ if and only if $pm(\bigcup_{j\leq i}\mathcal{T}_j) \models K\varphi$.*

Proof. First observe that if Σ_1 and Σ_2 are two disjoint vocabularies, φ is a formula over Σ_1, and Q_1 and Q_2 are *non-empty* possible world structures over Σ_1 and Σ_2 respectively, then $Q_1 \models K\varphi$ if and only if $Q_1 \oplus Q_2 \models K\varphi$.

Now if $\mathcal{T}$ is weakly permaconsistent, $D_{\mathcal{T}_i}(Q)$ is non-empty for all Q and i, i.e., all possible world structures used in the perfect model computation are non-empty. The result then follows from the above observation by induction. □

For "reasonable" theories (theories that represent the knowledge of a rational introspective agent), the inconsistent possible world structure will never be a relevant possible world structure. In this text, we focus on the class of weakly permaconsistent theories, i.e., those in which consistency is guaranteed. What we expect is that a good semantics for AEL manages to identify the intended possible world structure for "reasonable" monotonically stratified theories.

Definition 5.3.9 (Respect stratification)**.** We say that a semantics for autoepistemic logic *respects stratification* if all weakly permaconsistent monotonically stratified theories have exactly one model, namely their perfect model.

Following the analogy with logic programming, we would expect that stable and well-founded semantics respect stratification. However, this is not the case here! In fact, the motivating at the start of this section shows that well-founded semantics does not respect stratification. For (ultimate) stable semantics, this remains an open research question. In this chapter we will prove that grounded fixpoint semantics respects stratification.

Theorem 5.3.10. *The (ultimate) well-founded semantics for autoepistemic logic does not respect stratification.*

Proof. Example 5.3.1 provides a counterexample for the ultimate well-founded semantics. For the well-founded semantics based on another approximator, for example the one defined in (Denecker et al., 2003), this follows from the algebraical property that the ultimate well-founded fixpoint is always more precise than the A-well-founded fixpoint. □

Intuitively, the problem with the well-founded semantics is the fact that points in the bilattice, which are intervals of lattice elements, are not granular enough to represent the information in the informal reasoning process presented in Example. There, we would first like to restrict our attention to the set of possible world structures in which both $\neg Kp$ and $\neg K\neg p$ hold, i.e., no information about p is known. However, the smallest (most precise) interval that approximates this set is $[\bot, \top]$ (to see this, this interval must approximate $\{\{q\}, \{q, p\}\}$, $\{\{r\}, \{r, p\}\}$, $\{\{q, r\}, \{q, r, p\}\}$ and $\{\emptyset, \{p\}\}$). For this reason, well-founded inductions cannot leave the least precise bilattice point. In what follows, we define a refinement of the well-founded semantics that works on sets of lattice elements instead of intervals. This will allow us to follow the reasoning process of a rational agent in greater detail.

5.4 Set-Inductions

Given an operator $O : L \to L$, we now define two constructive characterisations of sets of elements of L and discuss how they relate to AFT. The first, and simplest, refines the Kripke-Kleene fixpoint. The Kripke-Kleene set, as we call it, manages to identify the intended model in Example 5.3.1 and for all monotonically stratified theories in which the stratification is strict. However, the semantics derived from it does not respect stratification in general. In order to overcome this limitation, the second constructive characterisation refines the well-founded semantics.

5.4.1 The Kripke-Kleene Set

We extend O to subsets of L in the pointwise manner. Below, X, X_i, Y denote subsets of L. Before defining our constructive characterisations, we recall how the ultimate Kripke-Kleene fixpoint is characterised. Given a set $X \subseteq L$, $[\![X]\!]$ denotes the smallest interval such that $X \subseteq [\![X]\!]$, i.e., $[\![X]\!] = [\bigwedge X, \bigvee X]$.

Lemma 5.4.1. *Let O be an operator. (x', y') is an U_O-application refinement of (x, y) if and only if*

$$[\![O([x, y])]\!] \subseteq [x', y'] \subseteq [x, y].$$

Proof. Trivial, since $U_O(x, y) = (\bigwedge O([x, y]), \bigvee O([x, y])) = [\![O([x, y])]\!]$. □

Now, the U_O-Kripke-Kleene fixpoint is characterised as the limit of any terminal monotone induction of U_O. In this computation, an interval $[x_i, y_i]$ is used to approximate the intended partial fixpoint. At each step, this interval is updated to the set $S_i := O([x_i, y_i])$, and then for the next step in the induction, the smallest interval approximating S_i is taken. With this information, it is clear how to generalise the Kripke-Kleene fixpoint to sets: all we need to do is skip the approximation by intervals, which might lose valuable information.

Definition 5.4.2 (Kripke-Kleene set-induction)**.** Let $O : L \to L$ be a lattice operator. A *Kripke-Kleene set-induction of O*, is a sequence $(X_i)_{i \leq \alpha}$ satisfying

- $X_0 = L$,
- $X_i \subseteq X_{i+1} \subseteq O(X_i)$, and
- $X_\lambda = \bigcap\{X_i \mid i < \lambda\}$, for limit ordinals $\lambda \leq \alpha$.

A Kripke-Kleene set-induction is *terminal* if there exists no $X_{\alpha+1} \neq X_\alpha$ such that $(X_i)_{i \leq \alpha+1}$ is a Kripke-Kleene set-induction induction.

Lemma 5.4.3. *All terminal Kripke-Kleene set-inductions converge to the same set, which we call the* Kripke-Kleene set of O *and denote $kks(O)$.*

Proof. Follows immediately from the fact that Kripke-Kleene set-inductions are monotone inductions of the extension of O to 2^L. □

Proposition 5.4.4. *Let A be an approximator of O and (x, y) be the A-Kripke-Kleene fixpoint of O. It then holds that $kks(O) \subseteq [x, y]$.*

Proof. Follows from the fact that the U_O-Kripke-Kleene fixpoint is more precise than the A-Kripke-Kleene fixpoint, Lemma 5.4.1 and the definition of $kks(O)$. □

Proposition 5.4.5. *The Kripke-Kleene set of O contains all fixpoints of O.*

Proof. Follows by induction from the fact that if X_i contains a fixpoint x of O, that then $O(X_i)$ also contains this fixpoint. □

The previous two propositions show that the Kripke-Kleene set refines the Kripke-Kleene fixpoint (for every approximator), but still approximates all fixpoints of O. For the example that motivated this work, the Kripke-Kleene set turns out to be precise enough.

Example 5.4.6 (Example 5.3.1 continued)**.** Again take

$$\mathcal{T} = \{q \Leftrightarrow \neg Kp, r \Leftrightarrow \neg Kq\}.$$

The construction of the Kripke-Kleene set of $D_\mathcal{T}$ starts from the entire lattice $\mathcal{W}_\Sigma$. We make two observations.

1. For every possible world structure Q, $D_\mathcal{T}(Q) \models K\neg q$ if $Q \models Kp$ and $D_\mathcal{T}(Q) \models Kq$ otherwise, and analogously for Kr.
2. As p does not occur objectively in $\mathcal{T}$, all possible world structures Q in the image of $D_\mathcal{T}$ satisfy that $I \in Q$ iff $I \cup \{p\} \in Q$.

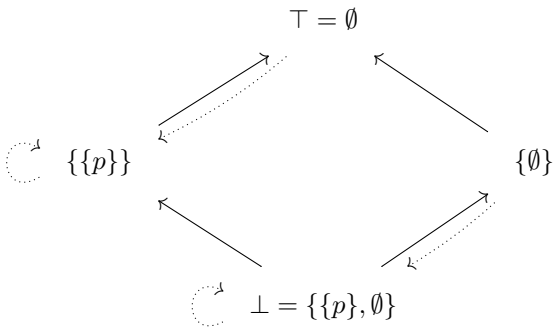
The above observations guarantee that $D_\mathcal{T}(\mathcal{W}_\Sigma) = \{Q, Q_q, Q_r, Q_{qr}\}$ where

- $Q \stackrel{\text{def}}{=} \{\{p\}, \{\}\}$,
- $Q_q \stackrel{\text{def}}{=} \{\{p, q\}, \{q\}\}$,
- $Q_r \stackrel{\text{def}}{=} \{\{p, r\}, \{r\}\}$, and
- $Q_{qr} \stackrel{\text{def}}{=} \{\{p, q, r\}, \{q, r\}\}$.

Furthermore, $D_\mathcal{T}(Q) = Q_{qr}$, $D_\mathcal{T}(Q_q) = Q_q$, $D_\mathcal{T}(Q_r) = Q_{qr}$, and $D_\mathcal{T}(Q_{qr}) = Q_q$. Thus $D^2_\mathcal{T}(\mathcal{W}_\Sigma) = \{Q_q, Q_{qr}\}$ and $D^3_\mathcal{T}(\mathcal{W}_\Sigma) = \{Q_q\}$. Thus, the Kripke-Kleene set of $D_\mathcal{T}$ is a singleton and the unique element of $kks(D_\mathcal{T})$ is the perfect model of $\mathcal{T}$.

In the next section we will show that it is not a coincidence that this theory has a singleton Kripke-Kleene set. A monotonically stratified theory in which the stratification is *strict* always has a singleton Kripke-Kleene set. This result does not hold for arbitrary monotonically stratified theories (if the stratification is not strict).

Example 5.4.7. The theory $\mathcal{T} = \{Kp \Rightarrow p\}$ is monotonically stratified with perfect model $\bot = \{\{p\}, \emptyset\}$. Its semantic operator is visualised below.



The Kripke-Kleene set of $D_{\mathcal{T}}$ is $\{\bot, \{\{p\}\}\}$, and hence is not a singleton.

5.4.2 The Well-Founded Set

In the previous section, we found that the Kripke-Kleene set sometimes is not precise enough. The reason why it fails is because it has no means to eliminate ungrounded reasoning. E.g., in Example 5.4.7, it fails to discover that knowledge of p only follows from the theory $Kp \Rightarrow p$ if p is known in the first place. Hence, a rational agent whose only knowledge is $\mathcal{T}$ would never know p. The well-founded semantics has a tool to eliminate this kind of reasoning, namely unfoundedness refinements. We generalise this kind of refinements to set inductions, but first, we recall some properties of ultimate well-founded inductions.

Lemma 5.4.8. *Let $O : L \to L$ be an operator and U_O its ultimate approximator. Suppose $(x_i, y_i)_{i \leq \beta}$ is an U_O-well-founded induction. For every $i < \beta$, one of the following conditions is satisfied:*

1. $[\![O([x_i, y_i])]\!] \subseteq [x_{i+1}, y_{i+1}] \subseteq [x_i, y_i]$, *or*
2. $[x_i, y_i] \setminus [x_{i+1}, y_{i+1}]$ *is a set of ungrounded points and* $O([x_{i+1}, y_{i+1}]) \subseteq [x_{i+1}, y_{i+1}]$.

Proof. If $(x_i, y_i) \to (x_{i+1}, y_{i+1})$ is an U_O-application refinement ((1) in Definition 2.2.1), then

$$\begin{aligned}[\![O([x_i, y_i])]\!] &= (\bigwedge O([x_i, y_i]), \bigvee O([x_i, y_i])) \\ &= U_O(x_i, y_i) \\ &\subseteq [x_{i+1}, y_{i+1}] \\ &\subseteq [x_i, y_i].\end{aligned}$$

In this case (1) is satisfied.

On the other hand, if $(x_i, y_i) \to (x_{i+1}, y_{i+1})$ is an U_O-unfoundedness refinement ((2) in Definition 2.2.1),then it follows from Theorem 3.3.4 that no grounded fixpoints are removed. Furthermore, Denecker and Vennekens (2007) (Proposition 2) showed that it always holds that $U_O(x_{i+1}, y_{i+1}) \subseteq [x_{i+1}, y_{i+1}]$; we then indeed find $O([x_{i+1}, y_{i+1}]) \subseteq U_O(x_{i+1}, y_{i+1}) \subseteq [x_{i+1}, y_{i+1}]$. □

This lemma inspires a generalised notion of refinement that works on sets instead of intervals and is based on the same intuitions.

Definition 5.4.9 (Refinement)**.** An *O-refinement* of X is a set Y such that one of the following holds:

- $O(X) \subseteq Y \subseteq X$, or
- $Y = X \setminus U$, where U is a set of ungrounded lattice elements and $O(Y) \subseteq Y$.

Following the analogy with A-refinements, we call the first type of refinement *application refinement* and the second kind *unfoundedness refinement*. An O-refinement is *strict* if $X \neq Y$.

If Y is an O-refinement of X and O is clear from the context, we often denote this $X \to Y$.

Definition 5.4.10 (Well-founded set-induction)**.** Let β be an ordinal number. A *well-founded set-induction* (or shortly, set-induction) is a sequence $L = X_0 \to X_1 \to \cdots \to X_\beta$ such that the following hold:

- For limit ordinals λ, $X_\lambda = \bigcap(\{X_\alpha \mid \alpha < \lambda\}$.
- For every $i < \beta$, $X_i \to X_{i+1}$ is a strict O-refinement.

Definition 5.4.11 (Terminal set)**.** A set-induction $(X_i)_{i\leq\beta}$ is *terminal* if X_β has no strict O-refinements. We call Y a *terminal set* of O if there is a terminal set-induction $(X_i)_{i\leq\beta}$ with $X_\beta = Y$.

In what follows, we show that terminal sets exist and also, that the terminal set is unique. Thus, even though in general many different set-inductions of an operator might exist, they all converge to the same set. An essential property is that refinements preserve being closed under O. This property is similar to A-contractingness (Denecker and Vennekens, 2007).

Proposition 5.4.12. *Given a set-induction $(X_i)_{i\leq\beta}$, for all $i \leq \beta$, it holds that $O(X_i) \subseteq X_i$.*

Proof. We prove this by induction. It is true initially since $X_0 = L$.

This property is preserved in limit ordinals λ since

$$O\left(\bigcap\{X_\alpha \mid \alpha < \lambda\}\right) \subseteq \bigcap\{O(X_\alpha) \mid \alpha < \lambda\}$$
$$\subseteq \bigcap\{X_\alpha \mid \alpha < \lambda\}.$$

Let $X \to Y$ be an O-refinement and suppose $O(X) \subseteq X$. We show that $O(Y) \subseteq Y$. If $X \to Y$ is an application refinement, then $O(X) \subseteq Y \subseteq X$. In this case also $O(Y) \subseteq O(X) \subseteq Y$. If $X \to Y$ is an unfoundedness refinement, then the result is trivial. This indeed shows that being closed under application of O is preserved by O-refinements. □

Proposition 5.4.13. *Every well-founded induction $(X_i)_{i\leq\beta}$ is decreasing, i.e. for $i < j$, it holds that $X_j \subseteq X_i$.*

Proof. It is trivial that unfoundedness refinements result in a decreased set and that taking limits results in a decreased set. Proposition 5.4.12 shows that also application refinements result in a decreased set. The result follows by induction. □

Proposition 5.4.14. *Every operator O has a terminal set.*

Proof. Follows from Proposition 5.4.13: we can extend every non-terminal induction with a strict O-refinement. This results in a strictly decreasing sequence of subsets of 2^L, which must eventually terminate since 2^L is a complete lattice. □

Proposition 5.4.15. *Let Z be a terminal set of O and let $X \to Y$ be an O-refinement. If $Z \subseteq X$, then $Z \subseteq Y$.*

Proof. Suppose Z is a terminal set of O. By proposition 5.4.13, it holds that $O(Z) \subseteq Z$. Since Z is terminal $O(Z) \not\subset Z$, otherwise $O(Z)$ would be a refinement of Z. Hence, we know that $Z = O(Z)$.

If $X \to Y$ is an application-refinement, then $O(X) \subseteq Y$, hence $Z = O(Z) \subseteq O(X) \subseteq Y$.

On the other hand, suppose $Y = X \setminus U$, where U are ungrounded points. Define $Z' = Z \setminus U$. It then holds that $Z' = Z \setminus U \subseteq X \setminus U = Y$. Also, Z' is a refinement of Z, since all elements in $Z' \setminus Z$ are ungrounded and

$$O(Z') \subseteq O(Y) \cap O(Z) \subseteq Y \cap Z = Z'.$$

Since Z is a terminal set, this refinement cannot be strict, hence, $U \cap Z = \emptyset$ and $Z \subseteq Y$. □

Theorem 5.4.16. *All set-inductions converge to the same set which we call the* well-founded set of O *and denote $wfs(O)$.*

Proof. Let $(X_i)_{i \leq \beta}$ and $(Y_i)_{i \leq \alpha}$ be terminal set-inductions. We show that $X_\beta = Y_\alpha$. From Propositions 5.4.15, it follows using induction that $X_\beta \subseteq Y_i$ for every $i \leq \alpha$. Hence, also $X_\beta \subseteq Y_\alpha$. The exact same argument can also be used to show that $X_\beta \supseteq Y_\alpha$, hence the result follows. □

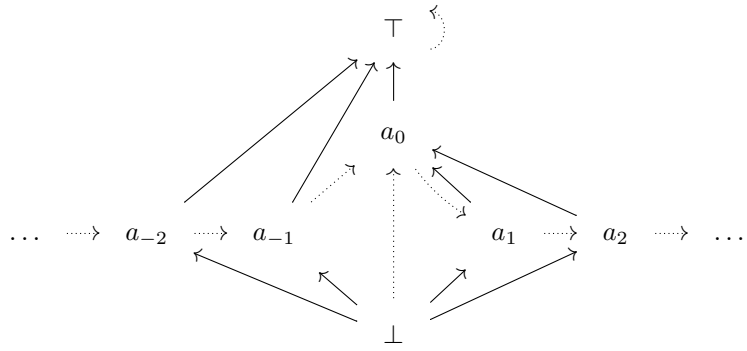
In AFT, it is known that the A-well-founded fixpoint is more precise than the A-Kripke-Kleene fixpoint for every approximator A. A similar result holds between the Kripke-Kleene set and the well-founded set.

Proposition 5.4.17. *For every operator O, $wfs(O) \subseteq kks(O)$.*

Proof. It follows directly from the definition that Kripke-Kleene set-inductions are exactly the well-founded set inductions that only use application refinements. The result then immediately follows from the fact that set-inductions are decreasing (Proposition 5.4.13). □

Example 5.4.18. Consider a lattice $\{\bot, \top\} \cup \{a_i \mid i \in \mathbb{Z}\}$ with order as depicted below and an operator O that maps that maps $\bot$ to a_0, $\top$ to $\top$ and

every a_i to a_{i+1}.



First, we notice that $\{\bot\} \cup \{a_i \mid i \geq 0\}$ are the O-grounded points. Indeed, $\top$ is ungrounded since $O(\top \wedge a_0) \leq a_0$ and for $i < 0$ it holds that $O(a_i \wedge a_0) = O(\bot) = a_0 \leq a_0$. We also notice that $\top$ is the only fixpoint of O. Hence, O has no grounded fixpoints. We now show that the well-founded set of O is empty. Let $X_0 = L$ and $X_1 = O(L) = L \setminus \{\bot\}$. Then, $O(X_1) = X_1$, hence no application refinements are possible. The observations above guarantee that $U = \{a_i \mid i < 0\} \cup \{\top\}$ is a set of ungrounded points. Let X_2 denote $X_1 \setminus U = \{a_i \mid i \geq 0\}$. Then $O(X_2) \subseteq X_2$ and we can refine X_1 to X_2. We notice that $O(X_2) \neq X_2$ hence more application refinements are possible. Unfoundedness refinement has now removed all ungrounded elements in the $\{a_i \mid i \in \mathbb{Z}\}$. More application refinements will gradually delete every of the remaining elements: $X_i = \{a_j \mid j \geq i - 2\}$ for $i > 2$. Hence, this example also illustrates that well-founded inductions can lead to an empty set.

Definition 5.4.19 (Total operator)**.** We call O *total* if its well-founded set is a singleton.

Intuitively, total operators are "good" in the sense that they uniquely determine one lattice point of interest.

Example 5.4.20 (Example 5.4.6 continued)**.** We saw that the semantic operator of

$$\mathcal{T} = \{q \Leftrightarrow \neg Kp, r \Leftrightarrow \neg Kq\}$$

has a singleton Kripke-Kleene set. Hence, there are no application refinements of $kks(D_\mathcal{T})$. Since the unique element $\{\{p, q\}, \{q\}\}$ of $kks(D_\mathcal{T})$ is a grounded fixpoint of $D_\mathcal{T}$, there are no unfoundedness refinements possible either and it follows that $wfs(D_\mathcal{T}) = kks(D_\mathcal{T})$. Hence $D_\mathcal{T}$ is total.

Example 5.4.21 (Example 5.4.7 continued)**.** With $\mathcal{T} = \{Kp \Rightarrow p\}$, we already know that the Kripke-Kleene set of $D_\mathcal{T}$ is $\{\bot, \{\{p\}\}\}$. As $\{\{p\}\}$ is not grounded for $D_\mathcal{T}$, a set-induction further refines this set to $\{\bot\}$. This is the well-founded set of $D_\mathcal{T}$.

In Section 5.5, we show that it is not a coincidence that the operators in Examples 5.4.20 and 5.4.21 are total; there, we show that the semantic operator of a monotonically stratified theory is always total. First, we summarise how the well-founded set relates to other fixpoints studied in AFT.

Theorem 5.4.22. *For every operator O and consistent approximator A of O, let $wf(A)$ denote the A-well-founded fixpoint of O and $gf(O)$ the set of grounded fixpoints of O. It holds that*

$$gf(O) \subseteq wfs(O) \subseteq wf(A).$$

Furthermore, each of these inclusions can be strict.

Proof. The first inclusion is proven as Proposition 5.4.23 and the second as Proposition 5.4.26. In Examples 5.4.24 and 5.4.6, respectively, we show that these inclusion can be strict. □

The following propositions and example all prove parts of this result.

Proposition 5.4.23. *The well-founded set of O contains all grounded fixpoints of O.*

Proof. Grounded fixpoints belong to X_0 and are preserved in limit ordinals. As for refinements $X \to Y$, all fixpoints are preserved by applying O, and all grounded points are trivially preserved by unfoundedness refinements. □

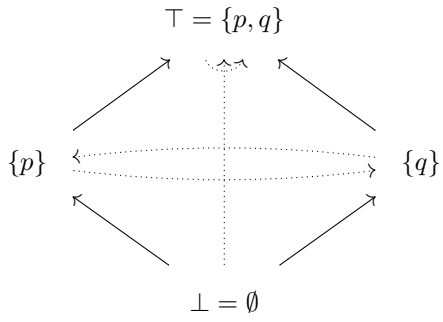
Example 5.4.24. Let L be the lattice $\{\bot, \top\}$ and O the operator that maps $\bot$ to $\top$ and $\top$ to $\bot$.

O has no fixpoints, hence also no grounded fixpoints. There are clearly no strict application refinements of L. Furthermore, since $\bot$ is grounded, there are no strict unfoundedness refinements either. Hence, it holds that $L = wfs(O)$. We notice that the well-founded set can be strictly larger than the set of grounded fixpoints of O.

Example 5.4.25. Consider the logic program $\mathcal{P}$

$$\left\{ \begin{array}{l} p \leftarrow \neg p \vee q. \\ q \leftarrow \neg q \vee p. \end{array} \right\}$$

The operator $T_{\mathcal{P}}$ is pictured below



We see that $T_{\mathcal{P}}(L) = \{\top, \{p\}, \{q\}\}$ consists of only grounded points and hence equals the well-founded set of O. Since not each of those are fixpoints, we again see that the well-founded set can be strictly larger than the set of grounded fixpoints of O.

Proposition 5.4.26. *The ultimate well-founded fixpoint of an operator O approximates the well-founded set of O.*

Proof. This follows immediately from Lemma 5.4.8, which shows that U_O-refinements are also O-refinements. □

Corollary 5.4.27. *Let A be a consistent approximator of O, then the A-well-founded fixpoint of O approximates the well-founded set of O.*

5.5 Locally Monotone Operators

In this section, we generalise the notion of monotonically stratified theories to the algebraical setting. We define the class of *locally monotone* operators and show that all operators in this class are total. In autoepistemic logic (AEL), a monotonically stratified theory has the property that whenever all symbols from a lower stratum are fixed, the semantic operator on the next stratum is monotone. Intuitively[2], we generalise this using equivalence relations. We

[2] All concepts used in these intuitions are formally defined below.

will call an operator locally monotone if there exists a sequence of equivalence relations $\equiv_i$ on L of increasing precision, such that within every equivalence class of $\equiv_i$, the quotient operator of O modulo $\equiv_{i+1}$ is monotone. The intended fixpoint of such an operator can then be computed by an iterated least fixpoint computation. Before formally introducing locally monotone operators, we first introduce some terminology on equivalence relations on lattices.

5.5.1 Meet Equivalences

An equivalence relation on a set is defined as usual. If $\equiv$ is an equivalence relation on a set X, we use $x_\equiv$ for the equivalence class $\{y \mid y \equiv x\}$ and if $Y \subseteq X$, $Y_\equiv$ denotes the set of equivalence classes of elements in Y. We define the quotient mapping $p_\equiv : X \to X_\equiv : x \mapsto x_\equiv$. If $x_\equiv$ is an equivalence class, every $y \in x_\equiv$ is a *representative* of $x_\equiv$. If E is a set of equivalence classes, a representative of E is any set Y such that $E = Y_\equiv$.

Definition 5.5.1 (Meet equivalence)**.** Let L be a complete lattice and $\equiv$ an equivalence relation on L. We call $\equiv$ a *meet equivalence* if $\equiv$ respects $\wedge$, i.e., if for every two subsets X and Y of L with $X_\equiv = Y_\equiv$, also $\bigwedge X \equiv \bigwedge Y$.

If $\equiv$ is a meet equivalence on L, then for every equivalence class $x_\equiv$, it holds that $\{x\} \equiv x_\equiv$, hence also $x = \bigwedge\{x\} \equiv \bigwedge x_\equiv$. This means that $x_\equiv$ has a least element $\bot_{x_\equiv} = \bigwedge x_\equiv$.

Definition 5.5.2 (Quotient order)**.** Let $\langle L, \leq\rangle$ be a complete lattice and $\equiv$ a meet equivalence on L. We define an order $\leq_\equiv$, called the *quotient order of L modulo* $\equiv$ on $L_\equiv$ as follows: $x_\equiv \leq_\equiv y_\equiv$ if $\bot_{x_\equiv} \leq \bot_{y_\equiv}$.

When $\equiv$ is clear from the context, we often abuse notation and omit $\equiv$, i.e., we use $\leq$ for the quotient order on $L_\equiv$.

Lemma 5.5.3. *Let $\langle L, \leq\rangle$ be a complete lattice and $\equiv$ a meet equivalence on L. Whenever $x \leq y$ for $x, y \in L$, it also holds that $x_\equiv \leq_\equiv y_\equiv$.*

Proof. If $x \leq y$, it holds that $x = x \wedge y$. Hence

$$\bot_{x_\equiv} \equiv x = x \wedge y \equiv \bot_{x_\equiv} \wedge \bot_{y_\equiv}.$$

Since $\bot_{x_\equiv} \equiv \bot_{x_\equiv} \wedge \bot_{y_\equiv}$ and $\bot_{x_\equiv}$ is the least element of $x_\equiv$, it must hold that $\bot_{x_\equiv} = \bot_{x_\equiv} \wedge \bot_{y_\equiv}$, i.e., that $\bot_{x_\equiv} \leq \bot_{y_\equiv}$. Thus, from the definition of $\leq_\equiv$, it then follows that $x_\equiv \leq_\equiv y_\equiv$. □

The order $\leq_{\equiv}$ defines a well-known concept, namely the quotient lattice. For completeness, we show (in Proposition 5.5.6) that this is indeed a complete lattice.

Lemma 5.5.4. *Let $\langle L, \leq \rangle$ be a complete lattice and $\equiv$ a meet equivalence on L. Let $X \subseteq L_{\equiv}$ be a set of equivalence classes and $X' \subseteq L$ be the set of least representatives of X, (i.e., $X' = \{\bot_x \mid x \in X\}$). Then $(\bigwedge X')_{\equiv}$ is a greatest lower bound of X in $L_{\equiv}$.*

Proof. First, we show that $(\bigwedge X')_{\equiv}$ is a lower bound of X. For every $x \in X$ it holds that $\bot_x \in X'$ hence also that

$$\bot_{(\bigwedge X')_{\equiv}} \leq \bot_x.$$

Hence, it also holds that $(\bigwedge X')_{\equiv} \leq_{\equiv} x$ for every $x \in X$.

Now we show that $(\bigwedge X')_{\equiv}$ is the greatest lower bound X. If y is another lower bound of X, then it must hold that $\bot_y \leq \bot_x$ for each $x \in X$. Hence, it must hold that $\bot_y \leq \bigwedge\{\bot_x \mid x \in X\}$, i.e., that $y \leq_{\equiv} (\bigwedge X')_{\equiv}$. □

Lemma 5.5.5. *Let $\langle L, \leq \rangle$ be a complete lattice and $\equiv$ a meet equivalence on L. Let $X \subseteq L_{\equiv}$ be a set of equivalence classes and $X' \subseteq L$ be the set of least representatives of X, (i.e., $X' = \{\bot_x \mid x \in X\}$). Then $(\bigvee X')_{\equiv}$ is a least upper bound of X in $L_{\equiv}$.*

Proof. First, we show that $(\bigvee X')_{\equiv}$ is an upper bound of X. For every $x \in X$ it holds that

$$\bot_x \leq \bigvee\{\bot_y \mid y \in X\} = \bigvee X',$$

hence by Lemma 5.5.3, for every $x \in X$, it holds that

$$x = (\bot_x)_{\equiv} \leq_{\equiv} \left(\bigvee X'\right)_{\equiv}.$$

Now we show that $(\bigvee X')_{\equiv}$ is the least upper bound of X. If y is another upper bound of X, then it must hold that $\bot_x \leq \bot_y$ for each $x \in X$, hence, it must hold that $\bigvee\{\bot_x \mid x \in X\} \leq \bot_y$. Again, by Lemma 5.5.3, it follows that $(\bigvee X')_{\equiv} \leq y$. □

Proposition 5.5.6. *Let $\langle L, \leq \rangle$ be a complete lattice and $\equiv$ a meet equivalence on L. Then $\langle L_{\equiv}, \leq_{\equiv} \rangle$ is a complete lattice, called the* quotient lattice of L modulo $\equiv$.

Proof. $L_{\equiv}$ is clearly a partially ordered set. Lemmas 5.5.4 and 5.5.5 guarantee that $\langle L_{\equiv}, \leq_{\equiv} \rangle$ permits least upper bounds and greatest lower bounds.

□

It follows easily from the definition of meet equivalence that the following lemma holds.

Lemma 5.5.7. *If $\equiv$ is a meet equivalence on L, then (i) every chain of representatives of $x_{\equiv}$ has a greatest lower bound in $x_{\equiv}$ and (ii) whenever $x_{\equiv} \leq_{\equiv} y_{\equiv}$, there is a $z \in x_{\equiv}$ such that $z \leq y$.*

Proof. For the first point, we note that if $X \subseteq x_{\equiv}$, then $\bigwedge X \equiv \bigwedge\{x\}$ by the definition of meet equivalence. Hence indeed $\bigwedge X \in x_{\equiv}$. For the second point, we can take $z = \bot_{x_{\equiv}}$. □

We would like to use equivalence relations as a tool to reason in a stratified way. For example in 5.3.1, the first observation was: p does not occur objectively in $\mathcal{T}$, hence in the intended model, it is unknown whether or not p holds. This observation was obtained by reasoning modulo the equivalence relation $Q \equiv Q'$ if $Q|_{\{p\}} = Q'|_{\{p\}}$, i.e., by only reasoning about knowledge of p. After these observations, we focused our attention on the set of possible world structures Q such that $Q \models \neg Kp$ and $Q \models \neg K\neg p$. From the situation in AEL, we can learn several properties that good equivalence relations (equivalence relations that generalise projecting out a variable in AEL) need to satisfy.

In order to illustrate these properties, we depicted the lattice $\mathcal{W}_{\{p,q\}}$ and the equivalence relation that corresponds to "forgetting q", i.e., the relation $\equiv$ such that $Q \equiv Q'$ if and only if $Q|_{\{p\}} = Q'|_{\{p\}}$ in Figure 5.1. First, we notice that the depicted equivalence relation in AEL is a meet equivalence but not a join equivalence (it does not preserve joins: for instance, $\{\emptyset, \{p\}\} \equiv \{\emptyset, \{p, q\}\}$ but $\{\emptyset, \{p\}\} \vee \{\emptyset, \{p, q\}\} = \{\emptyset\} \not\equiv \{\emptyset, \{p\}\}$). Second, simply being a meet equivalence is insufficient to generalise monotonically stratified theories. The definition of perfect model, in order to compute a least fixpoint, requires that every class $x_{\equiv}$ is chain complete: in the definition of perfect model, we take least upper bounds of a chain *within* equivalence classes (for computing the least fixpoint of D_i). Lemma 5.5.7 shows that for meet equivalences, greatest lower bounds can be taken within equivalence classes, but this does not hold for least upper bounds. Third, we observe that in AEL, the equivalence classes are nicely structured in the sense that whenever $x_{\equiv} \leq y_{\equiv}$, there is a $z \in y_{\equiv}$ such that $x \leq z$.

Our observations in AEL complement the observations in Lemma 5.5.7. In order to obtain these properties in general, we introduce the notion of a *conservative*

meet equivalence. In Section 5.5.2, we will use this to generalise monotonically stratified theories to operators.

Definition 5.5.8 (Conservative)**.** Let L be a complete lattice and $\equiv$ a meet equivalence on L. We call $\equiv$ *conservative* if *(i)* each class $x_\equiv$ is chain complete and *(ii)* whenever $x_\equiv \leq y_\equiv$, there is a $z \in y_\equiv$ with $x \leq z$.

We can order equivalence relations according to how fine-grained they are: one equivalence relation is finer than another if its equivalence classes are smaller; formally $\equiv_2$ is *finer* than $\equiv_1$ (denoted $\equiv_1 \leq_f \equiv_2$) if for all $x, y \in L$ with $x \equiv_2 y$, also $x \equiv_1 y$.

Proposition 5.5.9. *If $\equiv_1$ and $\equiv_2$ are conservative meet equivalences of L with $\equiv_1 \leq_f \equiv_2$, then $\equiv_1$ defines an equivalence relation on the lattice $L_{\equiv_2}$. This is a conservative meet equivalence of $L_{\equiv_2}$.*

Proof. $\equiv_1$ defines an equivalence relation on $L_{\equiv_2}$ as follows: $x_{\equiv_2} \equiv_1 y_{\equiv_2}$ if and only if $x \equiv_1 y$. This equivalence relation is well-defined (independent of the choice of representatives) since $\equiv_2$ is finer than $\equiv_1$, The fact that this is a meet equivalence follows directly from the fact that it is a meet equivalence on L. We now show that it is conservative.

First, assume that C is a chain of elements of $L_{\equiv_2}$, i.e., a totally ordered subset of $L_{\equiv_2}$ such that for every $c, c' \in C$, it holds that $c \equiv_1 c'$. Since $\equiv_2$ is a conservative meet equivalence of L, we can choose representatives $d_c in L$ for every $c \in C$ such that $d_c \leq d_{c'}$ if $c \leq c'$. Since all $c \in C$ are equivalent modulo $\equiv_1$, and the d_c are representatives of elements in C, it also holds $d_c \equiv_1 d_{c'}$ if $c, c' \in C$. Thus, $D := \{d_c \mid c \in C\}$ is a chain in L. Let d denote $\bigvee D$. Since $\equiv_1$ is conservative, $d \equiv_1 d_c$ for every c. Hence $d_{\equiv_2}$ is a least upper bound of C as desired.

Second, if $x, y \in L_{\equiv_2}$ and $x_{\equiv_1} \leq y_{\equiv_1}$, choose a representative $x' \in L$ of x with respect to $\equiv_2$. Since $\equiv_1 \leq_f \equiv_2$, it also holds that $x'_{\equiv_1} \leq y_{\equiv_1}$. Hence, we find a $z \in L$ such that $z_{\equiv_1} = y_{\equiv_1}$ and such that $x' \leq z$ since $\equiv_1$ is a conservative meet equivalence on L. Then $x \leq z_{\equiv_2}$, as desired. □

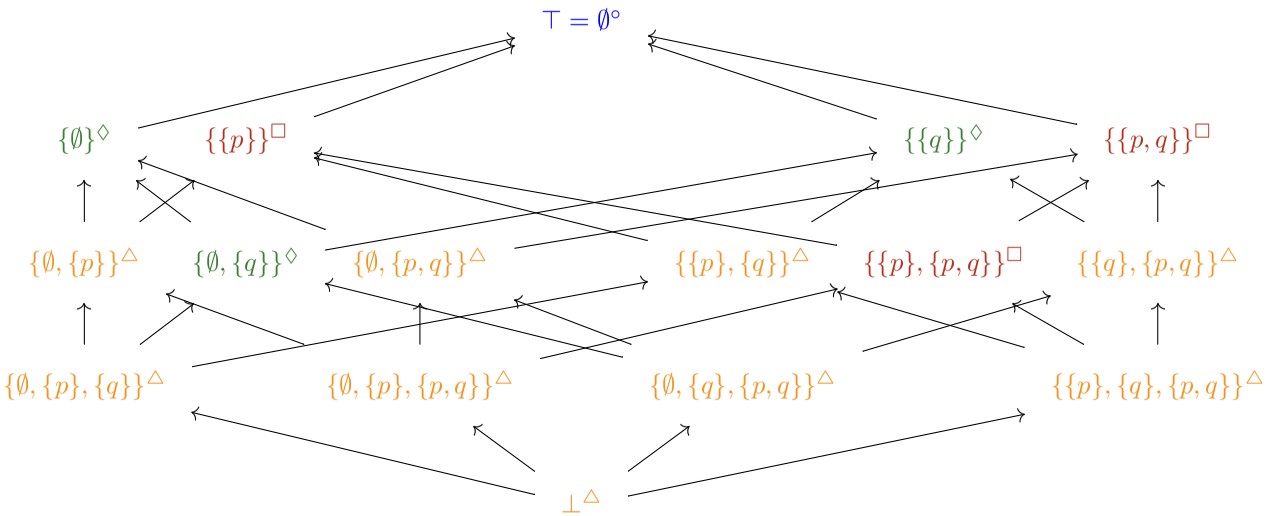


Figure 5.1: The lattice $\mathcal{W}_\Sigma$ for $\Sigma = \{p, q\}$. The colours and symbols represent the equivalence classes of lattice elements where two elements are equivalent if they have the same knowledge about p and $\neg p$. The orange elements (annotated with $\triangle$) are those in which both $\neg Kp$ and $\neg K\neg p$ hold. The green ($\Diamond$) elements satisfy $K\neg p$ and $\neg Kp$, the red ($\Box$) satisfy Kp and $\neg K\neg p$ and the blue ($\circ$) Kp and $K\neg p$.

5.5.2 Equivalences and Operators

Definition 5.5.10 (Congruence)**.** Let L be a complete lattice and $\equiv$ an equivalence relation on L and $O : L \to L$ an operator on L. We call $\equiv$ a *congruence* of O if for every $x, y \in L$, $x \equiv y$ implies $O(x) \equiv O(y)$.

The following lemma rephrases a well-known result about congruences.

Lemma 5.5.11. *If $\equiv$ is a congruence of O , then O defines a unique operator, denoted $O_{\equiv}$, on $L_{\equiv}$ such that $p_{\equiv} \circ O = O_{\equiv} \circ p_{\equiv}$.*

Proof. Any operator $O_{\equiv}$ satisfying the above condition must map $x_{\equiv} \in L_{\equiv}$ to $O(x)_{\equiv}$, hence uniqueness follows. This mapping is well-defined (independent of the choice of the representative x) since $\equiv$ is a congruence of O . □

Definition 5.5.12 (Abstraction/Specialisation)**.** If $L_{\equiv}$ is a conservative meet equivalence on L, and $\equiv$ is a congruence of O , then we call $O_{\equiv}$ an *abstraction* of O and O a *specialisation* of $O_{\equiv}$.

Example 5.5.13. Let $\Sigma = \{p, q\}$ be a vocabulary and $\Sigma_p = \{p\}$ a subvocabulary. Let $\mathcal{P}$ and $\mathcal{P}'$ be the following logic programs

$$\mathcal{P} = \left\{ \begin{array}{l} p. \\ q \leftarrow \neg p. \end{array} \right\},$$

$$\mathcal{P}' = \left\{ \; p. \; \right\}.$$

$\mathcal{P}$ and $\mathcal{P}'$ induce immediate consequence operators $T_{\mathcal{P}}$ and $T_{\mathcal{P}'}$ on the lattice of Σ-structures and Σ_p-structures respectively. The relation $x \equiv y$ if $x|_{\Sigma_p} = y|_{\Sigma_p}$ is a meet equivalence on 2^{Σ}. The operator $T_{\mathcal{P}_{\equiv}}$ then exactly equals $T_{\mathcal{P}'}$. Hence, $T_{\mathcal{P}'}$ is an abstraction of $T_{\mathcal{P}}$, obtained by dropping all rules defining symbols not in Σ_p from $\mathcal{P}$.

For every lattice, the total binary relation is a conservative meet equivalence. Hence, every operator is a specialisation of the trivial operator on a singleton lattice.[3] This shows that specialisations do not necessarily preserve the structure of the operator. In order to guarantee that some structure is preserved, we again find inspiration in AEL, more concretely in Proposition 5.3.3 which states that if $(\mathcal{T}_i)_{0 \leq i \leq n}$ is a monotonic stratification of $\mathcal{T}$, then the operator $D_{\mathcal{T}}$ is monotone within equivalence classes.

[3] There is only one operator on a singleton lattice.

Definition 5.5.14 (≡-monotone/constant)**.** Let $\equiv$ be a congruence of O. We call O *$\equiv$-monotone* (respectively *$\equiv$-constant*) if O is monotone (respectively constant) within each of $\equiv$, i.e., if $O|_{x_\equiv}$ is monotone (respectively constant) for every equivalence class $x_\equiv \in L_\equiv$.

Definition 5.5.15 (Monotone/constant specialisation/abstraction)**.** A specialisation O of $O_\equiv$ is called *monotone* (respectively *constant*) if O is $\equiv$-monotone (respectively $\equiv$-constant). An abstraction $O_\equiv$ of O is called *monotone* (respectively *constant*) if the specialisation O of $O_\equiv$ is monotone (respectively constant).

Example 5.5.16 (Example 5.5.13 continued)**.** The abstraction defined in Example 5.5.13 is a constant (and hence also a monotone) abstraction. Intuitively, for logic programs, an abstraction obtained by projecting out some symbols is constant if the symbols that are projected out do not depend on themselves, and it is monotone if they only depend positively on themselves. In the example, the symbol q which is projected out only depends on p, not on itself.[4]

The following propositions and theorems show that monotone and constant specialisations preserve a lot of structure of the operator. Our first result is a study of the relationship between the grounded fixpoints of O and those of $O_\equiv$.

Proposition 5.5.17. *Suppose O is a monotone specialisation of $O_\equiv$. If x is grounded for O, then $x_\equiv$ is grounded for $O_\equiv$.*

Proof. We prove this by contraposition. Choose $x \in L$ such that $x_\equiv$ is not grounded for $O_\equiv$. We show that x is not grounded either. Let $x' = x_\equiv$. Since x' is not grounded, we can choose $v' \in L_\equiv$ such that $O_\equiv(x' \wedge v') \leq v'$ and $x' \not\leq v'$. For every $v \in p_\equiv^{-1}(v')$, it holds that

$$\begin{aligned} p_\equiv(O(x \wedge v)) &= O_\equiv(p_\equiv(x \wedge v)) \\ &= O_\equiv(p_\equiv(x) \wedge p_\equiv(v)) \\ &= O_\equiv(x' \wedge v') \\ &\leq v'. \end{aligned}$$

Hence, using the definition of conservative meet equivalence, for every $v \in p_\equiv^{-1}(v')$, the set $\{w \in p_\equiv^{-1}(v') \mid w \geq O(x \wedge v)\}$ is non-empty. Since $p_\equiv$ is a preserves meets, the operator

$$\lambda : p_\equiv^{-1}(v') \to p_\equiv^{-1}(v') : v \mapsto \bigwedge \{w \in p_\equiv^{-1}(v') \mid w \geq O(x \wedge v)\}$$

[4] In Section 5.7, we formally define dependencies.

is well-defined. We know that O is monotone within every equivalence class. Hence, in particular, O is monotone on $x \wedge p_{\equiv}^{-1}(v')$. Since also $x \wedge \cdot$ and $\bigwedge$ are monotone, we find that λ, the composition of these three is monotone as well. Hence, it is a monotone operator in a chain complete subset of L, thus it has a least fixpoint, which we denote v. Since $\lambda(v) = v$ it holds that

$$v = \bigwedge \{w \in p_{\equiv}^{-1}(v') \mid w \geq O(x \wedge v)\}$$

and in particular

$$v \geq O(x \wedge v).$$

Furthermore $x \not\leq v$ since $p_{\equiv}(x) = x' \not\leq v' = p_{\equiv}(v)$. Hence indeed, x is not grounded. □

Example 5.5.18. The converse of Proposition 5.5.17 does not hold. Consider a simple lattice $L = \{\bot, \top\}$ with identity operator

$$O : L \to L : x \mapsto x.$$

Since O is monotone, O is a monotone specialisation of the trivial operator on a singleton lattice (using the equivalence relation $\bot \equiv \top$). However, $\top_{\equiv}$ is grounded while $\top$ is not.

The converse of Proposition 5.5.17 does not hold as the previous example shows. However, a better correspondence holds: Theorem 5.5.20 shows that there is a one-to-one correspondence between the grounded fixpoints of O and those of $O_{\equiv}$.

Proposition 5.5.19. *Let O be a monotone specialisation of $O_{\equiv}$. If x' is a grounded fixpoint of $O_{\equiv}$, then $\mathrm{lfp}(O|_{p_{\equiv}^{-1}(x')})$ is a grounded fixpoint of O.*

Proof. Suppose $x \stackrel{\text{def}}{=} \mathrm{lfp}(O|_{p_{\equiv}^{-1}(x')})$ is not grounded. Then there is a v such that $O(x \wedge v) \leq v$ and $x \not\leq v$. Let v' denote $v_{\equiv}$. Then also $O_{\equiv}(x' \wedge v') \leq v'$, and since x' is grounded, $x' \leq v'$. Hence, there exists a w with $v \equiv w$ such that $x \leq w$. Since $x \wedge w = x$ and $v \equiv w$, it holds that $x \wedge v \equiv x$. Now, we find that $O(x \wedge v) \leq O(x) = x$. Since also $O(x \wedge v) \leq v$, $x \wedge v$ is a prefixpoint of the monotone operator $O|_{p_{\equiv}^{-1}(x')}$ which is smaller than the least fixpoint of that same operator. This yields a contradiction. □

Theorem 5.5.20. *Let O be a monotone specialisation of $O_{\equiv}$. The mapping $p_{\equiv}$ induces a one-to-one correspondence between the grounded fixpoints of O and the grounded fixpoints of $O_{\equiv}$.*

Proof. Proposition 5.5.17 guarantees that $p_{\equiv}$ maps grounded points to grounded points. By definition of $O_{\equiv}$, it is clear that $p_{\equiv}$ also maps fixpoints to fixpoints. Now, Proposition 5.5.19 guarantees that for every grounded fixpoint x' of $O_{\equiv}$, there is a grounded fixpoint $x = \mathrm{lfp}(O|_{p_{\equiv}^{-1}(x')})$ of O with $p_{\equiv}(x) = x'$. Furthermore, since O is monotone on $p_{\equiv}^{-1}(x')$, all other fixpoints in $p_{\equiv}^{-1}(x')$ cannot be grounded as they are not minimal. □

The second main result, which we prove below, is that whenever the Kripke-Kleene set of an operator identifies a unique point of interest, then so do *constant* specialisations. This is not necessarily the case for *monotone* specialisations.

Proposition 5.5.21. *Let O be a monotone specialisation of $O_{\equiv}$. Then $kks(O) \subseteq p_{\equiv}^{-1}(kks(O_{\equiv}))$.*

Proof. Follows by induction from the fact that $O(p_{\equiv}^{-1}(X')) \subseteq p_{\equiv}^{-1}(O_{\equiv}(X'))$ for every $X' \subseteq L_{\equiv}$. □

Theorem 5.5.22. *If O is a constant specialisation of $O_{\equiv}$ and $kks(O_{\equiv})$ is a singleton, then $kks(O)$ is a singleton as well.*

Proof. Let $kks(O_{\equiv}) = \{x'\}$ be the Kripke-Kleene set of $O_{\equiv}$. Let $X = p_{\equiv}^{-1}(x')$. Proposition 5.5.21 guarantees that $kks(O) \subseteq X$. Since x' is a fixpoint of $O_{\equiv}$, $O(X) \subseteq X$. But, as $O|_X$ is constant, $O(X)$ is a singleton set containing a fixpoint of O, and hence the Kripke-Kleene set of O is indeed a singleton as well. □

Example 5.5.23. Consider an autoepistemic theory

$$\mathcal{T} = \{Kp \Rightarrow p\};$$

the semantic operator $D_{\mathcal{T}}$ is monotone and hence a monotone specialisation of the trivial operator on the singleton lattice. However, the Kripke-Kleene set of $D_{\mathcal{T}}$ is not a singleton. This shows that Theorem 5.5.22 does not necessarily hold for *monotone* specialisations.

Our third main result is that monotone specialisations preserve totality.

Proposition 5.5.24. *Suppose O is a monotone specialisation of $O_{\equiv}$ and $X' \to Y'$ is an $O_{\equiv}$-refinement, then $p_{\equiv}^{-1}(X') \to p_{\equiv}^{-1}(Y')$ is an O-refinement of the same kind.*

Proof. In this proof, we use X and Y to denote $p_{\equiv}^{-1}(X')$ and $p_{\equiv}^{-1}(Y')$ respectively.

If $O_{\equiv}(X') \subseteq Y' \subseteq X'$, then also

$$p_{\equiv}^{-1}(O_{\equiv}(X')) \subseteq p_{\equiv}^{-1}(Y') = Y \subseteq p_{\equiv}^{-1}(X') = X.$$

The result then follows from the fact that

$$O(X) = O(p_{\equiv}^{-1}(X')) \subseteq p_{\equiv}^{-1}(O_{\equiv}(X')).$$

On the other hand, suppose $X' \setminus Y'$ consists of ungrounded points and $O_{\equiv}(Y') \subseteq Y'$. Then by Proposition 5.5.17 also $X \setminus Y$ consists of ungrounded points. Also, for every $y \in Y$, $O_{\equiv}(p_{\equiv}(y)) \in Y'$, hence $O(Y) \subseteq Y$. □

Proposition 5.5.25. *Let O be a monotone specialisation of $O_{\equiv}$. Then $wfs(O) \subseteq p_{\equiv}^{-1}(wfs(O_{\equiv}))$.*

Proof. Let $(X'_i)_{i \leq \beta}$ be a terminal set-induction of $O_{\equiv}$. By iterated application of Proposition 5.5.24, $(p_{\equiv}^{-1}(X'_i))_{i \leq \beta}$ is a (not necessarily terminal) set-induction of O. Hence, the results follows. □

Theorem 5.5.26. *If O is a monotone specialisation of $O_{\equiv}$ and $O_{\equiv}$ is total, then O is total as well.*

Proof. Let $wfs(O_{\equiv}) = \{x'\}$ be the well-founded set of $O_{\equiv}$ and $x \stackrel{\text{def}}{=} \mathrm{lfp}(O|_{p_{\equiv}^{-1}(x')})$. We claim that $wfs(O) = \{x\}$.

Proposition 5.5.19 shows that x is a grounded fixpoint of O, thus using Proposition 5.4.23 we find that $x \in wfs(O)$. Proposition 5.5.25 guarantees that $wfs(O) \subseteq p_{\equiv}^{-1}(x')$. Since O is monotone on $p_{\equiv}^{-1}(x')$, any well-founded induction that ends in an X with $\{x\} \subseteq X \subseteq p_{\equiv}^{-1}(x')$ can, by repeated application refinements be reduced to $\{y \in X \mid y \not\leq x\}$. Next, unfoundedness refinement with $v = x$ yields a well-founded induction ending in $\{x\}$. Since $x \in wfs(O)$, no more refinements are possible and this well-founded induction is terminal. Hence $wfs(O) = \{x\}$ as desired. □

A monotone specialisation corresponds in the case of AEL to adding one extra stratum to a monotonically stratified theory, where the symbols in the new stratum only depend positively on their own knowledge. It is now clear how to generalise monotonically stratified theories to the algebraical setting, namely as an operator that can be constructed by iteratively applying monotonic specialisations starting from the trivial operator. We call such operators *locally monotone* and use *locally constant* for the stricter definition where at every stage a constant specialisation is applied (in the context of AEL, we will see that this kind of operators is derived from strict monotonically stratified theories).

Definition 5.5.27 (Locally monotone, locally constant)**.** Let O be an operator. We call O *locally monotone* (respectively *locally constant*) if there exists a sequence $(\equiv_i)_{i \leq n}$ of conservative meet equivalences of L such that $\equiv_i \geq_f \equiv_j$ if $i \geq j$ and the following all hold:

- $\equiv_n$ is the identity relation, hence $O_{\equiv_n} = O$,
- $\equiv_0$ is the trivial relation: $\forall x, y \in L : x \equiv_0 y$,
- each $\equiv_i$ is a congruence of O,
- $O_{\equiv_{i+1}}$ is a monotone (respectively constant) specialisation of $O_{\equiv_i}$ for every $i < n$ (by Proposition 5.5.9 $\equiv_{i+1}$ is a conservative meet equivalence on $L_{\equiv_i}$).

The following theorems now follow by repeated application of Theorems 5.5.22 and 5.5.26 respectively. They show that locally monotone (and locally constant) operators are "good" in the sense that they determine a unique point of interest. As we will see, for prudent monotonically stratified theories, this unique point of interest is indeed the perfect model.

Theorem 5.5.28. *Locally constant operators have a singleton Kripke-Kleene set.*

Proof. The proof is by induction. If O is a locally constant operator and the $\equiv_i$ are as in Definition 5.5.27, then $O_{\equiv_0}$ trivially has a singleton Kripke-Kleene set. The induction step follows directly from Theorem 5.5.22. □

Theorem 5.5.29. *Locally monotone operators are total.*

Proof. The proof is by induction. If O is a locally monotone operator and the $\equiv_i$ are as in Definition 5.5.27, then $O_{\equiv_0}$ is trivially total. The induction step follows directly from Theorem 5.5.26. □

5.6 Locally Monotone Operators in Autoepistemic Logic

We now define two semantics for autoepistemic logic of which our algebraic results show that they respect stratification. The first one is based on the concept of a grounded fixpoint, as discussed in Section 3.6; the second is based on our refinement of the well-founded semantics.

Definition 5.6.1 (Grounded fixpoint semantics)**.** The *grounded fixpoint semantics* for AEL is defined by $Q \models_{gf} \mathcal{T}$ if and only if Q is a grounded fixpoint of $D_{\mathcal{T}}$.

Definition 5.6.2 (Well-founded set semantics)**.** The *well-founded set semantics* for AEL is defined by $Q \models_{wfs} \mathcal{T}$ if and only if $\{Q\} = wfs(D_{\mathcal{T}})$.

Our algebraical results immediately show that the well-founded set semantics is slightly more liberal than the two-valued (ultimate) well-founded semantics which states that Q is a model if and only if (Q, Q) is the (ultimate) well-founded fixpoint: we found that if (Q, Q) is the (ultimate) well-founded fixpoint, then $\{Q\}$ is the well-founded set. We now formalise the relationship between monotonically stratified theories and locally monotone operators, in order to show that the two aforementioned semantics respect stratification.

Theorem 5.6.3. *If $\mathcal{T}$ is a weakly permaconsistent monotonically stratified autoepistemic theory, then $D_{\mathcal{T}}$ is locally monotone. Furthermore, if the stratification is strict, then $D_{\mathcal{T}}$ is locally constant.*

Proof. Suppose $(\mathcal{T}_i)_{0 \leq i \leq n}$ is a prudent monotonic stratification of $\mathcal{T}$ with respect to the partition $(\Sigma_i)_{0 \leq i \leq n}$. Consider the equivalence relations $\equiv_i$ such that $Q \equiv_i Q'$ if and only if $Q|_{\cup_{j \leq i} \Sigma_j} = Q'|_{\cup_{j \leq i} \Sigma_j}$.

First, it follows directly from the definitions that each of the $\equiv_i$ is a conservative meet equivalence.

Second, each of these equivalences is a congruence of $D_{\mathcal{T}}$: for every interpretation I and possible world structure Q, it holds that $Q, I \models \mathcal{T}$ if and only if $Q, I \models \bigcup_{j \leq i} \mathcal{T}_j$ and $Q, I \models \bigcup_{j > i} \mathcal{T}_j$. For a fixed I, the first is completely determined by the restriction of Q to symbols in strata smaller than (or equal to) i. Furthermore, in $\bigcup_{j>i} \mathcal{T}_j$, the symbols of strata smaller than or equal to i do not occur objectively, hence the second does not influence $I|_{\cup_{j \leq i} \Sigma_j}$. We conclude that if $Q \equiv_i Q'$, also $D_{\mathcal{T}}(Q) \equiv_i D_{\mathcal{T}}(Q')$: $\equiv_i$ is indeed a congruence of $D_{\mathcal{T}}$.

Third, it is clear that for $i > j$, $\equiv_i \geq_f \equiv_j$.

Fourth, we claim that $(D_{\mathcal{T}})_{\equiv_i} = D_{\cup_{j \leq i} \mathcal{T}_j}$. This follows again from the observation that $Q, I \models \mathcal{T}$ if and only if $Q|_{\cup_{j \leq i} \Sigma_j}, I|_{\cup_{j \leq i} \Sigma_j} \models \bigcup_{j \leq i} \mathcal{T}_j$ and $Q|_{\cup_{j>i} \Sigma_j}, I|_{\cup_{j>i} \Sigma_j} \models \bigcup_{j>i} \mathcal{T}_j$. Since $\mathcal{T}$ is weakly permaconsistent, there is at least one such I for each Q, hence $D_{\mathcal{T}}(Q)$ is non-empty and for every i $D_{\mathcal{T}}(Q)|_{\cup_{j \leq i} \Sigma_j} = D_{\cup_{j \leq i} \mathcal{T}_j}(Q|_{\cup_{j \leq i} \Sigma_j})$, which proves our claim.

Fifth, the fact that each $(D_{\mathcal{T}})_{\equiv_{i+1}}$ is a monotone specialisation of $(D_{\mathcal{T}})_{\equiv_i}$ follows immediately from Proposition 5.3.3.

The five above observations combined indeed yield that $D_{\mathcal{T}}$ is locally monotone; the claim about strict stratifications follows completely analogously. □

Example 5.6.4. The converse of Theorem 5.6.3 does not hold. The theory $\{p \Leftrightarrow Kp\}$ is not monotonically stratified; however, its semantic operator is locally monotone and even locally constant. The possible world structure $\{\emptyset\}$, i.e., the world in which we know $\neg p$ is its perfect model, and also its unique model under grounded fixpoint and well-founded set semantics.

Corollary 5.6.5. *Both the grounded fixpoint semantics and the well-founded set semantics respect stratification.*

Proof. Theorem 5.6.3 yields that both of these semantics identify a unique fixpoint of interest. The fact that this indeed equals the perfect model of a monotonically stratified theory can be proven inductively. The base case, for monotone operators, is trivial. The induction step is given by Theorem 5.5.26. □

5.7 Locally Monotone Operators in Logic Programming

We now apply our abstract results in the context of logic programming.

For $p, q \in \Sigma$, we say that p *depends positively (respectively negatively)* on q (in $\mathcal{P}$) if q occurs in the scope of an even (respectively odd) number of negations in the body of some rule $r \in \mathcal{P}$ with $head(r) = p$. A logic program $\mathcal{P}$ is called *(locally) stratified*[5] (Przymusinski, 1988) if every atom in Σ can be assigned an ordinal number ω such that

- no atom depends on an atom with greater rank, and
- no atom depends negatively on an atom with the same rank.

As mentioned in the introduction, in this text, we focus on finite stratifications. In this case, for a locally stratified logic program, we can assign to every atom $p \in \Sigma$ a rank $rank(p) \in \{0, \dots, n\}$ for some $n \in \mathbb{N}$ such that the two above conditions are satisfied. We now show that $T_{\mathcal{P}}$ is locally monotone in this case. For every i, we define Σ_i as $\{p \in \Sigma \mid rank(p) \leq i\}$ and $\mathcal{P}_i$ as the logic program over Σ_i defined by $\mathcal{P}_i = \{r \in \mathcal{P} \mid head(r) \in \Sigma_i\}$. The following proposition

[5] Since we work in the propositional case, locally stratified logic programs and stratified logic programs are the same.

reformulates two well-known results about locally stratified logic programs in the lattice terminology used in this text.

Proposition 5.7.1. *If $\mathcal{P}$ is (locally) stratified, then the following hold.*

- *$T_{\mathcal{P}_0}$ is monotone,*
- *For every $i \leq n$, $T_{\mathcal{P}_{i+1}}$ is a monotone specialisation of $T_{\mathcal{P}_i}$.*

Proof. The fact that $T_{\mathcal{P}_0}$ is monotone follows immediately from the fact that all atoms of Σ_0 only occur positively in the body of rules in $T_{\mathcal{P}_0}$.

For every $i \leq n$, the equivalence relation $\equiv_i$ between two Σ-interpretations is defined as $I \equiv_i I'$ if $I \cap \Sigma_i = I' \cap \Sigma_i$. This is a conservative meet equivalence of 2^{Σ}. It is easy to see that each of the $\equiv_i$ is a congruence of $T_{\mathcal{P}}$: if two interpretations are identical on Σ_i, since atoms in Σ_i only depend on atoms in Σ_i, applying $T_{\mathcal{P}}$ will preserve this equivalence.

Furthermore, for each i, it holds that $(T_{\mathcal{P}})_{\equiv_i} = T_{\mathcal{P}_i}$. The fact that $T_{\mathcal{P}_{i+1}}$ is a monotone specialisation of $T_{\mathcal{P}_i}$ now follows immediately from the fact that atoms in Σ_{i+1} only depend positively on atoms in Σ_{i+1}. □

Theorem 5.7.2. *If $\mathcal{P}$ is (locally) stratified, then $T_{\mathcal{P}}$ is locally monotone.*

Proof. Follows immediately from Proposition 5.7.1. □

The following result immediately follows from Theorem 5.7.2, but this result could also have been obtained from the fact that locally stratified logic programs have an exact well-founded model.

Corollary 5.7.3. *If $\mathcal{P}$ is (locally) stratified, then $T_{\mathcal{P}}$ is a total operator.*

5.8 Related Work

Vennekens et al. (2006) have studied modularity of operators. They have results similar to ours, namely that certain fixpoints of interest (stable, well-founded) can be characterised as fixpoints of derived operators on an abstraction of the lattice. There are some key differences between our study and theirs.

First, their work focuses on *product lattices*, which are a special case of the equivalence relations we studied here: for product lattice $L = L_1 \times L_2$, the equivalence relation $(x_1, x_2) \equiv (y_1, y_2)$ iff $x_1 = x_2$ is a conservative meet

equivalence. In the context of autoepistemic logic, product lattices do not naturally arise. Vennekens et al. (2006) solved this problem by transforming lattices in AEL to a derived product lattice, but this transformation only preserves semantics for permaconsistent theories. All permaconsistent theories are weakly permaconsistent, but not necessarily the other way round, as witnessed by Example 5.3.1. Thus, we study a broader class of theories.

Second, we showed how to construct the unique grounded fixpoint for locally monotone operators stratum per stratum; they showed how well-founded and stable fixpoints can be obtained from the components of stratifiable operators on a product lattice. Our specialisations preserve grounded fixpoints and totality, but do not always preserve stable and well-founded fixpoints; this is good since this work was motivated by an example for which the well-founded semantics fails in the first place.

Third, our theory does not require extensions to the bilattice: all concepts are defined in terms of the original operator; no approximations are required.

Our work is closely related to the work by Niemelä (1991). He also defines a constructive semantics for autoepistemic logic with as main goal that the models (called *L-hierarchic expansions*) are "tightly grounded" in the theory.[6] It deserves to be noted that semantically, his approach works for Example 5.3.1: the unique L-hierarchic expansion is the intended model in that example. It is currently unknown whether the L-hierarchic expansion semantics respects stratification in general; researching this is a topic for future work. However, the constructive semantics by Niemelä (1991) is not enough for our purposes, as the proposed constructions do not follow the reasoning process of a rational introspective agent for several reasons. First of all, in Example 5.3.1, taking as *enumeration*[7] r, p, q, we would first derive $\neg Kr$, next $\neg Kp$ and finally Kq. I.e., we are able to derive $\neg Kr$ because we will *eventually* derive that q is known. Secondly, there is a lack of *confluence*: a theory such as

$$\{p \Leftrightarrow \neg Kq, q \Leftrightarrow \neg Kp\}$$

can have multiple "constructions" associated to it, resulting in a different final state. If this is the case, these constructions cannot represent the reasoning process of a non-schizophrenic rational agent. An important contribution that distinguishes our work from the previous is that we defined our constructive semantics *algebraically* in approximation fixpoint theory (AFT). As such, our results are not restricted to AEL. For example, locally monotone operators include both the immediate consequence operator of a locally stratified logic program and the semantic operator of a monotonically stratified autoepistemic theory.

[6] In that work, "grounded" is still an informal concept.

[7] For details and a definition of enumerations, we refer to (Niemelä, 1991).

5.9 Conclusion

In this chapter, we identified a problem with the well-founded semantics for AEL, namely that certain simple and intuitively clear theories have a three-valued well-founded model. We solved this problem by refining the well-founded semantics algebraically. We showed that a large class of lattice operators (called locally monotone operators) have a unique grounded fixpoint, and we provided a constructive characterisation of this fixpoint. This class of operators generalises monotonically stratified autoepistemic theories and (locally) stratified logic programs.

Chapter 6

Conclusion

6.1 Contributions

This dissertation started from the observation that similar constructs are defined in many different logics. These concepts often stem from the same intuitions, but are called differently. The main goal of this thesis was to identify such constructs and formalise them in a unifying framework, namely approximation fixpoint theory.

In Chapter 3, we focused on the notion of groundedness. Throughout history, for different logics, semantics were developed that allowed ungrounded, also called self-supporting models. This has led to a great deal of criticism and a significant amount of research was dedicated to the development of new semantics that avoid this kind of models. For example, in the case of logic programming, this has led to the development of perfect model semantics, stable semantics and well-founded semantics. In Chapter 3 we defined groundedness in an abstract, algebraical setting, namely AFT. We studied how grounded fixpoints related to other fixpoints studied in AFT and obtained the following major results

- all grounded fixpoints are minimal fixpoints (Proposition 3.2.9),
- all stable fixpoints are grounded (Proposition 3.3.1), and
- the well-founded fixpoint approximates all grounded fixpoints (Theorem 3.3.4).

We applied our theory to logic programming, autoepistemic logic, default logic, argumentation frameworks and abstract dialectical frameworks. In each of these domains, our algebraical results entail that several existing semantics (namely those induced by stable fixpoints or an exact well-founded fixpoint) are grounded, in the sense that they only accept grounded models. We found that our notion of groundedness indeed corresponds to existing intuitions in these domains. We also studied complexity of computing grounded models of logic programs.

This far, we showed that in each of the aforementioned domains, stable models are grounded and the well-founded model is grounded if it is two-valued. The next questions that naturally arise are "what about partial stable models?" and "what is the relation between the three-valued well-founded model and groundedness?". We answered these two questions in Section 4. In order to do this, we extended the notion of groundedness for a lattice operator to groundedness for an approximator. We obtained the following major algebraical results:

- all partial A-stable fixpoints are A-grounded (Proposition 4.2.4), and
- the A-well-founded fixpoint is the least precise A-grounded fixpoint (Corollary 4.2.8).

This again shows that existing semantics do a great job at avoiding ungrounded models. This also shows that the well-founded model is not *just* some grounded model: it is the least precise grounded model. As such, our definition of groundedness is now used to provide an alternative characterisation of an existing semantics. We applied the theory of partial grounded fixpoints to logic programming and found that in this context, (partial) grounded fixpoints are closely related to unfounded sets.

In Chapter 5, we tackled a different problem. This chapter starts from the observation that all previously defined constructive semantics for AEL fail to identify the unique possible world structure of interest for a very simple autoepistemic theory. The major results of this chapter are:

- we generalised the example to a class of autoepistemic theories, called monotonically stratified theories (Definition 5.3.2),
- we defined, algebraically, two new constructive semantics, namely the Kripke-Kleene-set and the well-founded set (Lemma 5.4.3 and Theorem 5.4.16),

- we defined a class of lattice operators, called locally monotone operators, from which we show that they have a singleton well-founded set (Definition 5.5.27 and Theorem 5.5.29),
- we showed that monotonically stratified theories induce a locally monotone operator, and hence that our new semantics is able to identify the unique model of interest (Theorem 5.6.3).

The notion of groundedness played a central role in defining the well-founded set.

6.2 Future Directions

We see two main future research directions: researching *applications* and *extensions* of AFT.

6.2.1 Applications of Approximation Fixpoint Theory

Approximation fixpoint theory is a highly general and abstract theory that unifies the semantics of different logics. We expect that the application domain of AFT reaches far beyond the domains where it is currently applied. An important future challenge is to identify these research domains. A priori, possible application domains include all fields in which some kind of fixpoint (construction) is used to define a semantics or a point of interest.

One particular application domain that comes to mind are nested least and greatest fixpoint definitions (Hou et al., 2010), a language closely related to the propositional μ-calculus (Kozen, 1983; Streett and Emerson, 1989). In this logic, a propositional *least (respectively greatest) fixpoint definition* Δ is an expression of the form

$$\lfloor \mathcal{R}, \Delta_1, \ldots, \Delta_n \rfloor \text{ respectively } \lceil \mathcal{R}, \Delta_1, \ldots, \Delta_n \rceil,$$

where $\mathcal{R}$ is a set of rules, the Δ_i are least or greatest fixpoint definitions themselves satisfying some more syntactical restrictions (for details, we refer to the work by Hou et al. (2010)).

For example the expression

$$\Delta := \left[\begin{array}{l} p \leftarrow q \\ \lfloor q \leftarrow p \rfloor \end{array} \right]$$

represents a recursive definition in which p is maximised, while q is minimised. Intuitively, the nesting determines the "importance" of maximising/minimising

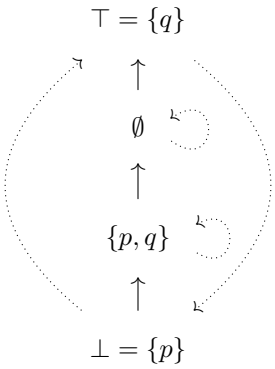
the truth value of certain atoms. In this example, maximising p (the outer fixpoint definition) is more important than minimising q (the inner fixpoint definition). Hence, the intended model of this definition is $\{p, q\}$.

This example has a natural translation to approximation fixpoint theory using the following observations

- we can construct an immediate consequence operator for Δ in the obvious way by ignoring all nesting information, i.e., by treating all rules as one rule set,
- in AFT, smaller points are always preferred over bigger points, hence a least fixpoint corresponds to the classical ordering of truth value while a greatest fixpoint corresponds to the reverse order,
- intuitively, the nesting information, i.e., the importance of the minimisation/maximisation of certain atoms can be encoded in lattices by means of a lexicographical order.

Formally, it is clear how to define the immediate consequence operator. We now explain how the lexicographical order should be constructed. For every definition, Σ_Δ denotes the set of atoms defined *locally* in Δ, i.e., those defined in Δ but not in its subdefinitions. Let Σ denote the entire vocabulary. Suppose I and J are Σ_Δ-interpretations, i.e., subsets of Σ_Δ, we say that $I \leq_\Delta J$ if either $I \subseteq J$ and Δ is a least fixpoint definition or if $I \supseteq J$ and Δ is a greatest fixpoint definition. If I and J are subsets of Σ, we say that $I \leq J$ if $I \cap \Sigma_\Delta \leq_\Delta J \cap \Sigma_\Delta$ or if $I \cap \Sigma_\Delta = J \cap \Sigma_\Delta$ and $I \leq_{\Delta_i} J$ for every subdefinition Δ_i of Δ.

Using this, our example reduces to the following lattice L and operator T_Δ:



Remarkably, in this example, the intended model is the unique stable fixpoint which is also the unique grounded fixpoint. This seems to be no coincidence,

as this holds for every example we have seen this far. It still remains an open question whether this holds in general. We formulate two conjectures.

Conjecture 6.2.1. *For every least or greatest fixpoint definition Δ and every interpretation I, it holds that $I \models \Delta$ if and only if I is the unique grounded fixpoint of T_Δ.*

Conjecture 6.2.2. *For every least or greatest fixpoint definition Δ and every interpretation I, it holds that $I \models \Delta$ if and only if I is the unique stable fixpoint of T_Δ.*

If one of these conjectures turns out to hold, the semantics of another formalism is captured by AFT.

Another observation is the fact that the well-founded fixpoint is $(\bot, \top)$. Hence, the well-founded fixpoint does not identify the intended fixpoint; neither does our refined construction: $wfs(T_\Delta) = \{\bot, \{p, q\}, \top\}$. On the other hand, Hou et al. (2010) did define a constructive characterisation of the intended fixpoint. It is an open research question whether their constructive characterisation can be generalised to arbitrary lattice operators.

6.2.2 Extensions of Approximation Fixpoint Theory

Another, but related, future research direction is further extending AFT. For example, in Section 3.4, we discussed how the notion of groundedness extends to programs with abstract constraint atoms. In that section, we restricted our attention to logic programs with abstract constraint atoms in the bodies of rules. Allowing them as well in heads gives rise to a *nondeterministic* generalisation of the immediate consequence operator. Some preliminary work has already been done to generalise semantics of AFT to nondeterministic operators, namely by Pelov and Truszczynski (2004), but several research questions remain in this area. For example, is it possible to provide a purely abstract algebraical characterisation of the stable semantics of a disjunctive logic program?

This dissertation adds another open question to this list. Namely, how does groundedness generalise to nondeterministic operators?

Appendix A

On Infinite Stratifications

In this chapter, we extend some of the theory from Chapter 5 to infinite stratifications. More concretely, we extend the notion of a locally monotone and locally constant operator to allow for a transfinite sequence of equivalence relations. We show that even in this generalised setting, it still holds that all locally constant operators have a singleton Kripke-Kleene set and all locally monotone operators are total.

A.1 Comparing Equivalences

We denote the set of all meet equivalences of L by $\mathcal{M}(L)$ and the set of all conservative meet equivalences of L by $\mathcal{C}(L)$. We can order equivalence relations according to how fine-grained they are: one equivalence relation is finer than another if its equivalence classes are smaller; we recall the following definition from Chapter 5.

Definition A.1.1 (Finer)**.** Let $\equiv_1$ and $\equiv_2$ be meet equivalences on L. We say that $\equiv_2$ is *finer* than $\equiv_1$ (denoted $\equiv_1 \leq_f \equiv_2$) if for all $x, y \in L$ with $x \equiv_2 y$, also $x \equiv_1 y$.

Proposition A.1.2. *For every subset $\mathfrak{A}$ of $\mathcal{M}(L)$, the relation $\equiv$ such that $x \equiv y$ if and only if $x \sim y$ for every $\sim$ in $\mathfrak{A}$ is a least upper bound of $\mathfrak{A}$ in $\langle \mathcal{M}(L), \leq_f \rangle$.*

Proof. It follows directly from the definitions that $\equiv$ is again a meet equivalence. Also, it is clear that $\sim \leq_f \equiv$ for every $\sim$ in $\mathfrak{A}$, and that $\equiv$ is the least relation with this property. □

It follows directly from the previous proposition that $\langle \mathcal{M}(L), \leq_f \rangle$ is chain complete. The finest meet equivalence of L is the identity relation, the least fine is the trivial equivalence relation in which everything is equivalent.

Proposition A.1.3. *The set $\mathcal{C}(L) \subseteq \mathcal{M}(L)$ is chain complete, i.e., the limit of a (possibly infinite) chain of conservative meet equivalences is again conservative.*

Proof. Let I be some (totally ordered) index set and $(\equiv_i)_{i\in I}$ a chain of conservative meet equivalences such that $\equiv_i \leq_f \equiv_j$ if $i \leq j$. Let $\equiv$ be $\bigvee_i \{\equiv_i\}$, we show that $\equiv$ is conservative.

First we prove that every equivalence class of $\equiv$ is chain complete. Let $x_\equiv$ be an equivalence class of $\equiv$ and let $C = (c_j)_{j\in J}$ be a chain in $x_\equiv$. Define $c = \bigvee_j c_j$; we show that $c \in x_\equiv$. First of all, it holds for every i and j that $c_j \in x_{\equiv_i}$, hence C is a chain in $x_{\equiv_i}$. Since $\equiv_i$ is conservative, $c \in x_{\equiv_i}$. Thus, using the definition of $\equiv$, it follows that $c \in x_\equiv$.

Secondly, we show that if $x_\equiv \leq y_\equiv$, then there is a $z \in y_\equiv$ such that $x \leq z$. Suppose $x_\equiv \leq y_\equiv$. For every i, it also holds that $x_{\equiv_i} \leq y_{\equiv_i}$, hence we find for every $i \in I$ a non-empty set $Z_i = \{z \in y_{\equiv_i} \mid z \geq x\}$. We define $z_i = \bigwedge Z_i$ for each i. If $i \leq j$, it holds that $\equiv_i \leq_f \equiv_j$, thus $Z_i \supseteq Z_j$ and $z_i \leq z_j$. It follows that $(z_i)_{i\in I}$ is a chain in L and its limit is in each of the $y_{\equiv_i}$. Hence, its limit is indeed an element of $y_\equiv$ greater than x. □

Proposition A.1.4. *Let $\mathfrak{A}$ be a set of meet equivalences of L. If $x_\equiv \leq y_\equiv$ for every $\equiv$ in $\mathfrak{A}$, then $x_{(\bigvee \mathfrak{A})} \leq y_{(\bigvee \mathfrak{A})}$.*

Proof. Let $\sim$ denote $\bigvee \mathfrak{A}$. It holds that $\bigvee_{\equiv\in\mathfrak{A}} \bot_{z_\equiv} = \bot_{z_\sim}$ for every $z \in L$. Also, for every $\equiv$ in $\mathfrak{A}$, it holds that $\bot_{x_\equiv} \leq \bot_{y_\equiv}$ since $x_\equiv \leq y_\equiv$. Hence we find that $\bot_{x_\sim} = \bigvee_{\equiv\in\mathfrak{A}} \bot_{x_\equiv} \leq \bigvee_{\equiv\in\mathfrak{A}} \bot_{y_\equiv} = \bot_{y_\sim}$. Thus indeed $x_{(\bigvee \mathfrak{A})} \leq y_{(\bigvee \mathfrak{A})}$. □

A.2 Extended locally monotone operators

We now extend the notion of a locally monotone operator to allow the possibility of an infinite stratification. The definition trivially extends to this setting now that we know how to take limits of chains of conservative meet equivalences.

Definition A.2.1 (∞-locally monotone, ∞-locally constant)**.** Let O be an operator. We call O ∞-*locally monotone* (respectively ∞-*locally constant*) if there exists a sequence $(\equiv_i)_{i\leq\beta}$ of conservative meet equivalences of L such that $\equiv_i \geq_f \equiv_j$ if $i \geq j$ and the following all hold:

- $\equiv_n$ is the identity relation, hence $O_{\equiv_n} = O$,
- $\equiv_0$ is the trivial relation: $\forall x, y \in L : x \equiv_0 y$,
- each of the $\equiv_i$ is a congruence of O,
- $O_{\equiv_{i+1}}$ is a monotone (respectively constant) specialisation of $O_{\equiv_i}$ for every $i < \beta$,
- $\equiv_\lambda = \bigvee(\{\equiv_i |\ i < \lambda\})$ for limit ordinals $\lambda < \beta$.

In order to prove that locally monotone operators are total, most work is already done in Theorem 5.5.26. The only difficulty left are limit ordinals. We now discuss some properties of limits of equivalences.

Proposition A.2.2. *Let $(\equiv_i)_{i<\beta}$ be a sequence of conservative meet equivalences of L that are congruences of O. Let $\equiv$ be $\bigvee(\{\equiv_i |\ i < \beta\})$. Then $x_\equiv$ is a fixpoint of $O_\equiv$ if and only if for every $i < \beta$, $x_{\equiv_i}$ is a fixpoint of $O_{\equiv_i}$.*

Proof. First suppose $x_{\equiv_i}$ is a fixpoint of $O_{\equiv_i}$ for every i. Hence $O(x) \equiv_i x$, for every i. Now, the definition of $\bigvee(\{\equiv_i |\ i < \beta\})$ guarantees that $O(x) \equiv x$, as desired.

For the other direction, we know that $\equiv \geq_f \equiv_i$ for every i. Hence if $O(x) \equiv x$, also $O(x) \equiv_i x$ for every i. □

Proposition A.2.3. *Let $(\equiv_i)_{i<\beta}$ be a sequence of conservative meet equivalences of L that are congruences of O and such that $\equiv_i \geq_f \equiv_j$ whenever $i \geq j$. Let $\equiv$ be $\bigvee(\{\equiv_i |\ i < \beta\})$. If $x_{\equiv_i}$ is grounded for each of the $i < \beta$, then $x_\equiv$ is grounded for $O_\equiv$.*

Proof. Suppose that for each of the $i < \beta$, $x_{\equiv_i}$ is grounded for $O_{\equiv_i}$ and that for some v

$$O_\equiv(x_\equiv \wedge v_\equiv) \leq v_\equiv.$$

Proposition 5.5.9 guarantees that each of the $\equiv_i$ is also a conservative meet equivalence of $L_\equiv$. Hence certainly $O_{\equiv_i}(x_{\equiv_i} \wedge v_{\equiv_i}) \leq v_{\equiv_i}$ for every $i < \beta$. Since $x_{\equiv_i}$ is grounded for $O_{\equiv_i}$, this means that $x_{\equiv_i} \leq v_{\equiv_i}$ for each i. Hence Proposition A.1.4 yields that $x_\equiv \leq v_\equiv$. □

Proposition A.2.4. *Let $(\equiv_i)_{i<\beta}$ be a sequence of conservative meet equivalences of L that are congruences of O and such that $\equiv_i \geq_f \equiv_j$ whenever $i \geq j$. Let $\equiv$ be $\bigvee(\{\equiv_i | \ i < \beta\})$. If X and Y are sets such that for every i, $X_{\equiv_i} \to Y_{\equiv_i}$ is a $O_{\equiv_i}$-refinement of the first (respectively the second) kind, then $X_{\equiv} \to Y_{\equiv}$ is a refinement of the first (respectively the second) kind.*

Proof. First of all, we claim that if $A, B \subseteq L$ and $A_{\equiv_i} \subseteq B_{\equiv_i}$ for every i, then $A_{\equiv} \subseteq B_{\equiv}$. This claim simply follows from the definition of $\bigvee(\{\equiv_i | \ i < \beta\})$.

If $p_{\equiv_i}(O(X)) = O_{\equiv_i}(X_{\equiv_i}) \subseteq Y_{\equiv_i} \subseteq X_{\equiv_i}$ for every y, then by the previous claim, it also holds that $O_{\equiv}(X_{\equiv}) \subseteq Y_{\equiv} \subseteq X_{\equiv}$.

On the other hand, suppose $X_{\equiv_i} \setminus Y_{\equiv_i}$ consists of ungrounded points and $O_{\equiv_i}(Y_{\equiv_i}) \subseteq Y_{\equiv_i}$. Then Proposition 5.5.17 guarantees that also $X \setminus Y$ consists of ungrounded points. Also, for every $y \in Y$ and every i, $O_{\equiv_i}(y_{\equiv_i}) \in Y_{\equiv_i}$, hence by our first claim, $O(Y) \subseteq Y$. □

Proposition A.2.5. *Let O be an operator such that the conditions in Proposition A.2.4 are satisfied. For every $i < \beta$, $wfs(O_\beta) \subseteq \{x_{\equiv_\beta} \mid x_{\equiv_i} \in wfs(O_{\equiv_i})\}$.*

Proof. Follows immediately by induction using Propositions A.2.4 and 5.5.24. Notice that Proposition 5.5.24 guarantees that the precondition in Proposition A.2.4 that all refinements are of the same kind is satisfied. □

Proposition A.2.6. *Let O be an operator such that the conditions in Proposition A.2.4 are satisfied. For every $i < \beta$, $kks(O_\beta) \subseteq \{x_{\equiv_\beta} \mid x_{\equiv_i} \in kks(O_{\equiv_i})\}$.*

Proof. Follows immediately by induction using Proposition A.2.4. □

Theorem A.2.7. *All ∞-locally monotone operators are total.*

Proof. The proof is by induction. If O is a locally monotone operator and the $\equiv_i$ are as in Definition 5.5.27, then $O_{\equiv_0}$ is trivially total. The induction step follows directly from Theorem 5.5.26.

Let $(\equiv_i)_{i\leq\beta}$ be a sequence of equivalences of L satisfying the conditions in Definition 5.5.27. We use L_i for $L_{\equiv_i}$ and O_i for $O_{\equiv_i}$.

We prove this by induction on β. For $\beta = 0$, the result is trivial. If the results holds for i, Theorem 5.5.26 guarantees that it holds for $i + 1$ as well.

Let λ be a limit ordinal and suppose for all $i < \lambda$, O_i is total, we show that O_λ is total. First, we show that $wfs(O_\lambda)$ non-empty. For every $i < \lambda$, let $w_i \in L_i$

denote the unique element in the well-founded set of O_i and let $x_i \in L$ the minimal representative of w_i, i.e., $x_i = \bot_{w_i}$. Since $\equiv_j \geq_f \equiv_i$ for $j \geq i$, it holds that $x_j \in w_i$ if $j \geq i$. Hence, the sequence (x_i) is increasing. Let $x = \bigvee_i x_i$ be the limit of this sequence. Now $x_{\equiv_i} = w_i$, for every i, hence Propositions A.2.3 and A.2.2 yield that $x_{\equiv_\lambda}$ is a grounded fixpoint of O_λ. Hence $wfs(O_\lambda)$ is non-empty.

Now, Proposition A.2.5 yields that:

$$wfs(O_\lambda) \subseteq \bigcap_{i<\lambda}(\{x_{\equiv_\lambda} \mid x_{\equiv_i} = w_i\})$$

Hence, if $x'_{\equiv_\lambda} \in wfs(O_\lambda)$, then for every i, $x'_{\equiv_i} = w_i = x_{\equiv_i}$. Thus, using the definition of $\bigvee(\{\equiv_i \mid i < \lambda\})$, we find that $x_{\equiv_\lambda} = x'_{\equiv_\lambda}$. □

Theorem A.2.8. *All ∞-locally constant operators have a singleton Kripke-Kleene set.*

Proof. The argument is completely analogous to the proof of Theorem A.2.7. □

Bibliography

Abiteboul, S., Hull, R., Vianu, V., 1995. Foundations of Databases. Addison-Wesley.

Abiteboul, S., Simon, E., Vianu, V., 1990. Non-deterministic languages to express deterministic transformations. In: (PODS, 1990), pp. 218–229.
URL http://doi.acm.org/10.1145/298514.298575

Abiteboul, S., Vianu, V., 1991. Datalog extensions for database queries and updates. J. Comput. Syst. Sci. 43 (1), 62–124.
URL http://dx.doi.org/10.1016/0022-0000(91)90032-Z

Aczel, P., 1977. An introduction to inductive definitions. In: Barwise, J. (Ed.), Handbook of Mathematical Logic. North-Holland Publishing Company, pp. 739–782.

Afanasiev, L., Grust, T., Marx, M., Rittinger, J., Teubner, J., 2008. An inflationary fixed point operator in XQuery. In: ICDE. IEEE, pp. 1504–1506.
URL http://dx.doi.org/10.1109/ICDE.2008.4497604

Alviano, M., Calimeri, F., Charwat, G., Dao-Tran, M., Dodaro, C., Ianni, G., Krennwallner, T., Kronegger, M., Oetsch, J., Pfandler, A., Pührer, J., Redl, C., Ricca, F., Schneider, P., Schwengerer, M., Spendier, L. K., Wallner, J. P., Xiao, G., 2013. The fourth Answer Set Programming competition: Preliminary report. In: (Cabalar and Son, 2013), pp. 42–53.
URL http://dx.doi.org/10.1007/978-3-642-40564-8_5

Andres, B., Obermeier, P., Sabuncu, O., Schaub, T., Rajaratnam, D., 2013. ROSoClingo: A ROS package for ASP-based robot control.
URL http://arxiv.org/abs/1307.7398

Andrews, T., Blockeel, H., Bogaerts, B., Bruynooghe, M., Denecker, M., De Pooter, S., Macé, C., Ramon, J., Aug. 2012. Analyzing manuscript traditions using constraint-based data mining. In: First Workshop on

Combining Constraint Solving with Mining and Learning, Montpellier, France, 27 August 2012.
URL https://lirias.kuleuven.be/handle/123456789/352303

Antic, C., Eiter, T., Fink, M., 2013. Hex semantics via approximation fixpoint theory. In: (Cabalar and Son, 2013), pp. 102–115.
URL http://dx.doi.org/10.1007/978-3-642-40564-8_11

Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D., Patel-Schneider, P. F. (Eds.), 2003. The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, second edition, 2007.
URL http://www.cambridge.org/asia/catalogue/catalogue.asp?isbn=9780521876254

Balint, A., Belov, A., Heule, M. J., Matti, M. J., 2013. The 2013 international SAT competition.
URL http://satcompetition.org/2013/results.shtml

Baral, C., 2003. Knowledge Representation, Reasoning, and Declarative Problem Solving. Cambridge University Press, New York, NY, USA.
URL http://ebooks.cambridge.org/ebook.jsf?bid=CBO9780511543357

Baral, C., De Giacomo, G., Eiter, T. (Eds.), 2014. Principles of Knowledge Representation and Reasoning: Proceedings of the Fourteenth International Conference, KR 2014, Vienna, Austria, July 20-24, 2014. AAAI Press.
URL http://www.aaai.org/Library/KR/kr14contents.php

Baral, C., Subrahmanian, V., 1993. Dualities between alternative semantics for logic programming and nonmonotonic reasoning. Journal of Automated Reasoning 10 (3), 399–420.
URL http://dx.doi.org/10.1007/BF00881799

Barker, S., Boella, G., Gabbay, D. M., Genovese, V., 2014. Reasoning about delegation and revocation schemes in answer set programming. J. Log. Comput. 24 (1), 89–116.
URL http://dx.doi.org/10.1093/logcom/exs014

Bi, Y., You, J.-H., Feng, Z., 2014. A generalization of approximation fixpoint theory and application. In: Kontchakov, R., Mugnier, M.-L. (Eds.), Web Reasoning and Rule Systems. Vol. 8741 of LNCS. Springer International Publishing, pp. 45–59.
URL http://dx.doi.org/10.1007/978-3-319-11113-1_4

Bidoit, N., Ykhlef, M., 1998. Fixpoint calculus for querying semistructured data. In: Atzeni, P., Mendelzon, A. O., Mecca, G. (Eds.), WebDB. Vol. 1590 of LNCS. Springer, pp. 78–97.
URL http://dx.doi.org/10.1007/10704656_6

Bogaerts, B., Jansen, J., Bruynooghe, M., De Cat, B., Vennekens, J., Denecker, M., 7 2014a. Simulating dynamic systems using linear time calculus theories. TPLP 14, 477–492.
URL http://journals.cambridge.org/article_S1471068414000155

Bogaerts, B., Jansen, J., De Cat, B., Janssens, G., Bruynooghe, M., Denecker, M., 2014b. Meta-level representations in the IDP knowledge base system: Towards bootstrapping inference engine development. In: Mitchell, D., Denecker, M. (Eds.), Workshop on Logic and Search, 2014.
URL https://lirias.kuleuven.be/handle/123456789/459071

Bogaerts, B., Vennekens, J., Denecker, M., 2015a. Grounded fixpoints. In: Bonet, B., Koenig, S. (Eds.), Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA. AAAI Press, pp. 1453–1459.
URL http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9324

Bogaerts, B., Vennekens, J., Denecker, M., 2015b. Grounded fixpoints and their applications in knowledge representation. Artif. Intell. 224, 51–71.
URL http://dx.doi.org/10.1016/j.artint.2015.03.006

Bogaerts, B., Vennekens, J., Denecker, M., Van den Bussche, J., 2014c. FO(C): A knowledge representation language of causality. TPLP 14 (4-5-Online-Supplement), 60–69.
URL https://lirias.kuleuven.be/handle/123456789/459436

Bogaerts, B., Vennekens, J., Denecker, M., Van den Bussche, J., 2014d. FO(C) and related modelling paradigms. In: Konieczny, S., Tompits, H. (Eds.), NMR. No. RR-1843-14-01 in INFSYS. Institut Für Informationssyteme, pp. 90–96.
URL http://arxiv.org/abs/1404.6394

Bogaerts, B., Vennekens, J., Denecker, M., Van den Bussche, J., 2014e. Inference in the FO(C) modelling language. In: Schaub, T., Friedrich, G., O'Sullivan, B. (Eds.), ECAI 2014 - 21st European Conference on Artificial Intelligence, 18-22 August 2014, Prague, Czech Republic - Including Prestigious Applications of Intelligent Systems (PAIS 2014). IOS Press, pp. 111–116.
URL http://dx.doi.org/10.3233/978-1-61499-419-0-111

Bonatti, P. A., 1995. Autoepistemic logics as a unifying framework for the semantics of logic programs. J. Log. Program. 22 (2), 91–149.
URL http://dx.doi.org/10.1016/0743-1066(94)00022-X

Brewka, G., Strass, H., Ellmauthaler, S., Wallner, J. P., Woltran, S., 2013. Abstract dialectical frameworks revisited. In: Rossi, F. (Ed.), IJCAI

2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013. IJCAI/AAAI.
URL http://www.aaai.org/ocs/index.php/IJCAI/IJCAI13/paper/view/6551

Brewka, G., Woltran, S., 2010. Abstract dialectical frameworks. In: Lin, F., Sattler, U., Truszczyński, M. (Eds.), Principles of Knowledge Representation and Reasoning: Proceedings of the Twelfth International Conference, KR 2010, Toronto, Ontario, Canada, May 9-13, 2010. AAAI Press, pp. 102–111.
URL http://aaai.org/ocs/index.php/KR/KR2010/paper/view/1294

Bruynooghe, M., Blockeel, H., Bogaerts, B., De Cat, B., De Pooter, S., Jansen, J., Labarre, A., Ramon, J., Denecker, M., Verwer, S., 2015. Predicate logic as a modeling language: Modeling and solving some machine learning and data mining problems with IDP3. TPLP(in press).
URL https://lirias.kuleuven.be/handle/123456789/448838

Buchholz, W., Feferman, S., Pohlers, W., Sieg, W., 1981. Iterated Inductive Definitions and Subsystems of Analysis: Recent Proof-Theoretical Studies. Vol. 897 of Lecture Notes in Mathematics. Springer.

Cabalar, P., Son, T. C. (Eds.), 2013. Logic Programming and Nonmonotonic Reasoning, 12th International Conference, LPNMR 2013, Corunna, Spain, September 15-19, 2013. Proceedings. Vol. 8148 of LNCS. Springer.

Cadoli, M., Donini, F. M., 1997. A survey on knowledge compilation. AI Commun. 10 (3-4), 137–150.
URL http://iospress.metapress.com/content/0a5ejbx7dl6r07j8/

Calvanese, D., De Giacomo, G., Lenzerini, M., 1999. Reasoning in expressive description logics with fixpoints based on automata on infinite trees. In: Dean, T. (Ed.), IJCAI. Morgan Kaufmann, pp. 84–89.
URL http://ijcai.org/Past%20Proceedings/IJCAI-99-VOL-1/PDF/013.pdf

Caminada, M. W. A., Carnielli, W. A., Dunne, P. E., 2012. Semi-stable semantics. J. Log. Comput. 22 (5), 1207–1254.
URL http://dx.doi.org/10.1093/logcom/exr033

Castro, L. F., Warren, D. S., 2001. An environment for the exploration of non monotonic logic programs. In: Kusalik, A. J. (Ed.), Proceedings of the Eleventh Workshop on Logic Programming Environments (WLPE'01), Paphos, Cyprus, December 1, 2001.
URL http://arxiv.org/abs/cs.PL/0111049

Darwiche, A., Marquis, P., 2002. A knowledge compilation map. J. Artif. Intell. Res. (JAIR) 17, 229–264.
URL http://dx.doi.org/10.1613/jair.989

De Cat, B., Bogaerts, B., Bruynooghe, M., Denecker, M., 2014a. Predicate logic as a modelling language: The IDP system. CoRR abs/1401.6312.
URL http://arxiv.org/abs/1401.6312

De Cat, B., Bogaerts, B., Denecker, M., Sep. 2014b. MiniSAT(ID) for satisfiability checking and constraint solving.
URL https://lirias.kuleuven.be/handle/123456789/463884

De Cat, B., Bogaerts, B., Devriendt, J., Denecker, M., 2013. Model expansion in the presence of function symbols using constraint programming. In: 2013 IEEE 25th International Conference on Tools with Artificial Intelligence, Herndon, VA, USA, November 4-6, 2013. IEEE Computer Society, pp. 1068–1075.
URL http://dx.doi.org/10.1109/ICTAI.2013.159

Decroix, K., Butin, D., Jansen, J., Naessens, V., Dec. 2014. Inferring Accountability from Trust Perceptions. In: Prakash, A., Shyamasundar, R. K. (Eds.), Information Systems Security, ICISS 2014, Hyderabad, 16-20 December 2014. Springer-Verlag, pp. 69–88.
URL https://lirias.kuleuven.be/handle/123456789/461505

Denecker, M., 2012. The FO(·) knowledge base system project: An integration project (invited talk). In: ASPOCP.

Denecker, M., De Schreye, D., 1993. Justification semantics: A unifying framework for the semantics of logic programs. In: Pereira, L. M., Nerode, A. (Eds.), LPNMR. MIT Press, pp. 365–379.
URL https://lirias.kuleuven.be/handle/123456789/133075

Denecker, M., Marek, V., Truszczyński, M., July 26-30 1998. Fixpoint 3-valued semantics for autoepistemic logic. In: AAAI'98. MIT Press, Madison, Wisconsin, pp. 840–845.
URL http://www.aaai.org/Papers/AAAI/1998/AAAI98-119.pdf

Denecker, M., Marek, V., Truszczyński, M., 2000. Approximations, stable operators, well-founded fixpoints and applications in nonmonotonic reasoning. In: Minker, J. (Ed.), Logic-Based Artificial Intelligence. Vol. 597 of The Springer International Series in Engineering and Computer Science. Springer US, pp. 127–144.
URL http://dx.doi.org/10.1007/978-1-4615-1567-8_6

Denecker, M., Marek, V., Truszczyński, M., 2003. Uniform semantic treatment of default and autoepistemic logics. Artif. Intell. 143 (1), 79–122.
URL http://dx.doi.org/10.1016/S0004-3702(02)00293-X

Denecker, M., Marek, V., Truszczyński, M., Jul. 2004. Ultimate approximation and its application in nonmonotonic knowledge representation systems. Information and Computation 192 (1), 84–121.
URL https://lirias.kuleuven.be/handle/123456789/124562

Denecker, M., Marek, V., Truszczyński, M., 2011. Reiter's default logic is a logic of autoepistemic reasoning and a good one, too. In: Brewka, G., Marek, V., Truszczyński, M. (Eds.), Nonmonotonic Reasoning – Essays Celebrating Its 30th Anniversary. College Publications, pp. 111–144.
URL http://arxiv.org/abs/1108.3278

Denecker, M., Ternovska, E., Apr. 2008. A logic of nonmonotone inductive definitions. ACM Trans. Comput. Log. 9 (2), 14:1–14:52.
URL http://dx.doi.org/10.1145/1342991.1342998

Denecker, M., Vennekens, J., 2007. Well-founded semantics and the algebraic theory of non-monotone inductive definitions. In: Baral, C., Brewka, G., Schlipf, J. S. (Eds.), LPNMR. Vol. 4483 of LNCS. Springer, pp. 84–96.
URL http://dx.doi.org/10.1007/978-3-540-72200-7_9

Denecker, M., Vennekens, J., 2008. Building a knowledge base system for an integration of logic programming and classical logic. In: (García de la Banda and Pontelli, 2008), pp. 71–76.
URL http://dx.doi.org/10.1007/978-3-540-89982-2_12

Denecker, M., Vennekens, J., 2014. The well-founded semantics is the principle of inductive definition, revisited. In: (Baral et al., 2014), pp. 22–31.
URL https://lirias.kuleuven.be/handle/123456789/448356

Devriendt, J., Bogaerts, B., Bruynooghe, M., 2014. BreakIDGlucose: On the importance of row symmetry. In: Proceedings of the Fourth International Workshop on the Cross-Fertilization Between CSP and SAT (CSPSAT).
URL https://lirias.kuleuven.be/handle/123456789/456639

Devriendt, J., Bogaerts, B., De Cat, B., Denecker, M., Mears, C., 2012. Symmetry propagation: Improved dynamic symmetry breaking in SAT. In: IEEE 24th International Conference on Tools with Artificial Intelligence, ICTAI 2012, Athens, Greece, November 7-9, 2012. IEEE, pp. 49–56.
URL http://dx.doi.org/10.1109/ICTAI.2012.16

Dung, P. M., 1995. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. Artif.

Intell. 77 (2), 321 – 357.
URL http://dx.doi.org/10.1016/0004-3702(94)00041-X

Eén, N., Sörensson, N., 2003. Temporal induction by incremental SAT solving. Electr. Notes Theor. Comput. Sci. 89 (4).
URL http://dx.doi.org/10.1016/S1571-0661(05)82542-3

Eiter, T., Ianni, G., Krennwallner, T., 2009. Answer set programming: A primer. In: Tessaris, S., Franconi, E., Eiter, T., Gutierrez, C., Handschuh, S., Rousset, M., Schmidt, R. A. (Eds.), Reasoning Web. Semantic Technologies for Information Systems, 5th International Summer School 2009, Brixen-Bressanone, Italy, August 30 - September 4, 2009, Tutorial Lectures. Vol. 5689 of Lecture Notes in Computer Science. Springer, pp. 40–110.
URL http://dx.doi.org/10.1007/978-3-642-03754-2_2

Eiter, T., Leone, N., Saccà, D., 1997. On the partial semantics for disjunctive deductive databases. Ann. Math. Artif. Intell. 19 (1-2), 59–96.
URL http://dx.doi.org/10.1023/A:1018947420290

Faber, W., Pfeifer, G., Leone, N., 2011. Semantics and complexity of recursive aggregates in answer set programming. Artif. Intell. 175 (1), 278–298.
URL http://dx.doi.org/10.1016/j.artint.2010.04.002

Fagin, R., Halpern, J. Y., Moses, Y., Vardi, M. Y., 1995. Reasoning About Knowledge. MIT Press.
URL http://library.books24x7.com.libproxy.mit.edu/toc.asp?site=bbbga&bookid=7008

Feferman, S., 1970. Formal theories for transfinite iterations of generalised inductive definitions and some subsystems of analysis. In: Kino, A., Myhill, J., Vesley, R. (Eds.), Intuitionism and Proof theory. North Holland, pp. 303–326.

Ferraris, P., 2005. Answer sets for propositional theories. In: Proceedings of International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR). pp. 119–131.
URL http://dx.doi.org/10.1007/11546207_10

Fitting, M., 2002. Fixpoint semantics for logic programming — A survey. Theoretical Computer Science 278 (1-2), 25–51.
URL http://dx.doi.org/10.1016/S0304-3975(00)00330-3

García de la Banda, M., Pontelli, E. (Eds.), 2008. Logic Programming, 24th International Conference, ICLP 2008, Udine, Italy, December 9-13 2008, Proceedings. Vol. 5366 of LNCS. Springer.

Gebser, M., Kaminski, R., Kaufmann, B., Schaub, T., 2012. Answer Set Solving in Practice. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers.
URL dx.doi.org/10.2200/S00457ED1V01Y201211AIM019

Gelfond, M., Lifschitz, V., 1988. The stable model semantics for logic programming. In: Kowalski, R. A., Bowen, K. A. (Eds.), ICLP/SLP. MIT Press, pp. 1070–1080.
URL http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.24.6050

Gelfond, M., Zhang, Y., 2014. Vicious circle principle and logic programs with aggregates. TPLP 14 (4-5), 587–601.
URL http://dx.doi.org/10.1017/S1471068414000222

Gottlob, G., 1995. Translating default logic into standard autoepistemic logic. J. ACM 42 (4), 711–740.
URL http://dx.doi.org/10.1145/210332.210334

Hallnäs, L., 1991. Partial inductive definitions. Theor. Comput. Sci. 87 (1), 115–142.
URL http://dx.doi.org/10.1016/S0304-3975(06)80007-1

Halpern, J. Y., Moses, Y., 1985. Towards a theory of knowledge and ignorance: Preliminary report. In: Apt, K. R. (Ed.), Logics and Models of Concurrent Systems. Vol. 13 of NATO ASI Series. Springer Berlin Heidelberg, pp. 459–476.
URL http://dx.doi.org/10.1007/978-3-642-82453-1_16

Hou, P., De Cat, B., Denecker, M., 2010. FO(FD): Extending classical logic with rule-based fixpoint definitions. TPLP 10 (4-6), 581–596.
URL http://dx.doi.org/10.1017/S1471068410000293

Kleene, S. C., 1938. On notation for ordinal numbers. The Journal of Symbolic Logic 3 (4), 150–155.
URL http://www.jstor.org/stable/2267778

Konolige, K., 1988. On the relation between default and autoepistemic logic. Artif. Intell. 35, 343–382.
URL http://dx.doi.org/10.1016/0004-3702(88)90021-5

Kozen, D., 1983. Results on the propositional μ-calculus. Theoretical Computer Science 27, 333–354.

Kreisel, G., 1963. Generalized inductive definitions. Tech. rep., Section III in the Stanford University report on the Foundations of Analysis.

Leone, N., Rullo, P., Scarcello, F., 1997. Disjunctive stable models: Unfounded sets, fixpoint semantics, and computation. Inf. Comput. 135 (2), 69–112.
URL http://dx.doi.org/10.1006/inco.1997.2630

Levesque, H. J., 1990. All I know: A study in autoepistemic logic. Artif. Intell. 42 (2-3), 263–309.
URL http://dx.doi.org/10.1016/0004-3702(90)90056-6

Levesque, H. J., Pirri, F., Reiter, R., 1998. Foundations for the situation calculus. Electron. Trans. Artif. Intell. 2, 159–178.
URL http://dblp.uni-trier.de/db/journals/etai/etai2.html#LevesquePR98

Lifschitz, V., 1999. Answer set planning. In: Schreye, D. D. (Ed.), Logic Programming: The 1999 International Conference, Las Cruces, New Mexico, USA, November 29 - December 4, 1999. MIT Press, pp. 23–37.

Lifschitz, V., 2008. Twelve definitions of a stable model. In: (García de la Banda and Pontelli, 2008), pp. 37–51.
URL http://dx.doi.org/10.1007/978-3-540-89982-2_8

Lin, F., Reiter, R., 1997. How to progress a database. Artif. Intell. 92 (1-2), 131–167.
URL http://dx.doi.org/10.1016/S0004-3702(96)00044-6

Marek, V., Niemelä, I., Truszczyński, M., 2008. Logic programs with monotone abstract constraint atoms. TPLP 8 (2), 167–199.
URL http://dx.doi.org/10.1017/S147106840700302X

Marek, V., Truszczynski, M., 1989. Relating autoepistemic and default logics. In: Brachman, R. J., Levesque, H. J., Reiter, R. (Eds.), Proceedings of the 1st International Conference on Principles of Knowledge Representation and Reasoning (KR'89). Toronto, Canada, May 15-18 1989. Morgan Kaufmann, pp. 276–288.
URL http://dl.acm.org/citation.cfm?id=112950

Marek, V., Truszczyński, M., 1991. Autoepistemic logic. J. ACM 38 (3), 588–619.
URL http://dx.doi.org/10.1145/116825.116836

Marek, V., Truszczyński, M., 1999. Stable models and an alternative logic programming paradigm. In: Apt, K. R., Marek, V., Truszczyński, M., Warren, D. S. (Eds.), The Logic Programming Paradigm: A 25-Year Perspective. Springer-Verlag, pp. 375–398.
URL http://arxiv.org/abs/cs.LO/9809032

Martin-Löf, P., 1971. Hauptsatz for the intuitionistic theory of iterated inductive definitions. In: Fenstad, J. (Ed.), Second Scandinavian Logic Symposium. pp. 179–216.

McDermott, D., 1982. Nonmonotonic logic II: Nonmonotonic modal theories. Journal of the ACM 29 (1), 33–57.
URL http://dl.acm.org/citation.cfm?id=322293

McDermott, D., Doyle, J., 1980. Nonmonotonic logic I. Artif. Intell. 13 (1-2), 41–72.
URL http://hdl.handle.net/1721.1/6303

Mitchell, D. G., Ternovska, E., 2005. A framework for representing and solving NP search problems. In: Veloso, M. M., Kambhampati, S. (Eds.), AAAI. AAAI Press / The MIT Press, pp. 430–435.
URL http://www.aaai.org/Library/AAAI/2005/aaai05-068.php

Moore, R. C., 1984. Possible-world semantics for autoepistemic logic. In: Proceedings of the Workshop on Non-Monotonic Reasoning. pp. 344–354, reprinted in: M. Ginsberg, ed., *Readings on Nonmonotonic Reasoning*, pages 137–142, Morgan Kaufmann, 1990.
URL http://www.sri.com/sites/default/files/uploads/publications/pdf/616.pdf

Moore, R. C., 1985. Semantical considerations on nonmonotonic logic. Artif. Intell. 25 (1), 75–94.
URL http://dx.doi.org/10.1016/0004-3702(85)90042-6

Moschovakis, Y. N., 1974a. Elementary Induction on Abstract Structures. North-Holland Publishing Company, Amsterdam- New York.

Moschovakis, Y. N., 1974b. On non-monotone inductive definability. Fundamenta Mathematica 82, 39–83.

Niemelä, I., 1991. Constructive tightly grounded autoepistemic reasoning. In: Mylopoulos, J., Reiter, R. (Eds.), Proceedings of the 12th International Joint Conference on Artificial Intelligence. Sydney, Australia, August 24-30, 1991. Morgan Kaufmann, pp. 399–405.

Niemelä, I., 1999. Logic programs with stable model semantics as a constraint programming paradigm. Ann. Math. Artif. Intell. 25 (3-4), 241–273.
URL http://dx.doi.org/10.1023/A:1018930122475

Papadimitriou, C. H., 1994. Computational complexity. Addison-Wesley.

Peirce, C., Buchler, J., 1955. Philosophical Writings of Peirce. International library of psychology, philosophy and scientific method. Dover Publications.
URL https://books.google.com/books?id=7KPdoAEACAAJ

Pelov, N., Denecker, M., Bruynooghe, M., 2007. Well-founded and stable semantics of logic programs with aggregates. TPLP 7 (3), 301–353.
URL http://dx.doi.org/10.1017/S1471068406002973

Pelov, N., Truszczynski, M., 2004. Semantics of disjunctive programs with monotone aggregates - an operator-based approach. In: Delgrande, J. P., Schaub, T. (Eds.), 10th International Workshop on Non-Monotonic Reasoning (NMR 2004), Whistler, Canada, June 6-8, 2004, Proceedings. pp. 327–334.
URL http://www.pims.math.ca/science/2004/NMR/papers/paper43.pdf

Pirri, F., Reiter, R., May 1999. Some contributions to the metatheory of the situation calculus. J. ACM 46 (3), 325–361.
URL http://doi.acm.org/10.1145/316542.316545

PODS, 1990. Proceedings of the Ninth ACM Symposium on Principles of Database Systems. ACM Press.

Post, E. L., 1943. Formal reductions of the general combinatorial decision problem. American Journal of Mathematics 65 (2), 197–215.
URL dx.doi.org/10.2307/2371809

Prokosch, H. U., Kamm, S., Wieczorek, D., Dudeck, J., 1991. Knowledge representation in pharmacology. A possible application area for the Arden syntax? Proc Annu Symp Comput Appl Med Care, 243–247.
URL http://www.ncbi.nlm.nih.gov/pubmed/1807597

Przymusinski, T. C., 1988. On the declarative semantics of deductive databases and logic programs. In: Foundations of Deductive Databases and Logic Programming. Morgan Kaufmann, pp. 193–216.

Przymusinski, T. C., 1991. Stable semantics for disjunctive programs. New Generation Computing 9 (3/4), 401–424.
URL http://dx.doi.org/10.1007/BF03037171

Randell, D. A., Cui, Z., Cohn, A. G., 1992. A spatial logic based on regions and connection. In: Nebel, B., Rich, C., Swartout, W. R. (Eds.), Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning (KR'92). Cambridge, MA, October 25-29, 1992. Morgan Kaufmann, pp. 165–176.

Reiter, R., 1980. A logic for default reasoning. Artif. Intell. 13 (1-2), 81–132.
URL http://dx.doi.org/10.1016/0004-3702(80)90014-4

Reiter, R., 2001. Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems. MIT Press.
URL http://books.google.be/books?id=exa4f6BOZdYC

Riazanov, A., Voronkov, A., 2002. The design and implementation of VAMPIRE. AI Communications 15 (2-3), 91–110.
URL http://iospress.metapress.com/content/ajar8kjbdtdf7kc2/

Saccà, D., 1997. The expressive powers of stable models for bound and unbound DATALOG queries. J. Comput. Syst. Sci. 54 (3), 441–464.
URL http://dx.doi.org/10.1006/jcss.1997.1446

Saccà, D., Zaniolo, C., 1997. Deterministic and non-deterministic stable models. J. Log. Comput. 7 (5), 555–579.
URL http://dx.doi.org/10.1093/logcom/7.5.555

Schlipf, J. S., 1995. Complexity and undecidability results for logic programming. Annals of Mathematics and Artificial Intelligence 15 (3-4), 257–288.
URL http://dx.doi.org/10.1007/BF01536398

Schnoebelen, Ph., 2003. The complexity of temporal logic model checking. In: Balbiani, Ph., Suzuki, N.-Y., Wolter, F., Zakharyaschev, M. (Eds.), Proceedings of the 4th Workshop on Advances in Modal Logic (AIML'02). King's College Publications, pp. 481–517.

Seipel, D., Minker, J., Ruiz, C., 1997. A characterization of the partial stable models for disjunctive databases. In: Maluszynski, J. (Ed.), Logic Programming, Proceedings of the 1997 International Symposium, Port Jefferson, Long Island, NY, USA, October 13-16, 1997. MIT Press, pp. 245–259.
URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.47.1

Shanahan, M., 1997. Solving the Frame Problem - a Mathematical Investigation of the Common Sense Law of Inertia. MIT Press.
URL http://mitpress.mit.edu/books/solving-frame-problem

Son, T. C., Pontelli, E., Elkabani, I., 2006. An unfolding-based semantics for logic programming with aggregates. CoRR abs/cs/0605038.
URL http://arxiv.org/abs/cs/0605038

Spector, C., 1961. Inductively defined sets of natural numbers. In: Infinitistic Methods (Proc. 1959 Symposium on Foundation of Mathematis in Warsaw). Pergamon Press, Oxford, pp. 97–102.

Strass, H., 2013. Approximating operators and semantics for abstract dialectical frameworks. Artif. Intell. 205, 39–70.
URL http://dx.doi.org/10.1016/j.artint.2013.09.004

Strass, H., Wallner, J. P., 2014. Analyzing the computational complexity of abstract dialectical frameworks via approximation fixpoint theory. In: (Baral et al., 2014), pp. 101–110.
URL http://www.aaai.org/ocs/index.php/KR/KR14/paper/view/7917

Streett, R. S., Emerson, E. A., 1989. An automata theoretic decision procedure for the propositional μ-calculus. Information and Computation 81 (3), 249–264.

Sutcliffe, G., 2013. The 6th IJCAR Automated Theorem Proving System Competition - CASC-J6. AI Communications 26 (2), 211–223.

Ternovska, E., Mitchell, D. G., 2009. Declarative programming of search problems with built-in arithmetic. In: Boutilier, C. (Ed.), IJCAI. pp. 942–947.
URL http://ijcai.org/papers09/Papers/IJCAI09-160.pdf

Thielscher, M., Jan. 2011. A unifying action calculus. Artif. Intell. 175 (1), 120–141.
URL http://dx.doi.org/10.1016/j.artint.2010.04.010

Truszczynski, M., 2006. Strong and uniform equivalence of nonmonotonic theories - an algebraic approach. Ann. Math. Artif. Intell. 48 (3-4), 245–265.
URL http://dx.doi.org/10.1007/s10472-007-9049-2

van Emden, M. H., Kowalski, R. A., 1976. The semantics of predicate logic as a programming language. J. ACM 23 (4), 733–742.
URL http://dx.doi.org/10.1145/321978.321991

Van Gelder, A., Ross, K. A., Schlipf, J. S., 1991. The well-founded semantics for general logic programs. J. ACM 38 (3), 620–650.
URL http://dx.doi.org/10.1145/116825.116838

van Harmelen, F., Lifschitz, V., Porter, B., 2007. Handbook of Knowledge Representation. Elsevier Science, San Diego, USA.
URL https://www.elsevier.com/books/handbook-of-knowledge-representation/van-harmelen/978-0-444-52211-5

Vardi, M. Y., 1986. Querying logical databases. Journal of Computer and System Sciences 33 (2), 142 – 160.
URL http://www.sciencedirect.com/science/article/pii/0022000086900164

Vennekens, J., Gilis, D., Denecker, M., 2006. Splitting an operator: Algebraic modularity results for logics with fixpoint semantics. ACM Trans. Comput. Log. 7 (4), 765–797.
URL http://dx.doi.org/10.1145/1182613.1189735

Vennekens, J., Gilis, D., Denecker, M., Jan. 2007. Erratum to splitting an operator: Algebraic modularity results for logics with fixpoint semantics (vol 7, pg 765, 2006).
URL https://lirias.kuleuven.be/handle/123456789/124415

Verheij, B., 1996. Two approaches to dialectical argumentation: Admissible sets and argumentation stages. In: In Proceedings of the biannual International Conference on Formal and Applied Practical Reasoning (FAPR) workshop. Universiteit, pp. 357–368.

Weidenbach, C., Dimova, D., Fietzke, A., Kumar, R., Suda, M., Wischnewski, P., 2009. SPASS version 3.5. In: Schmidt, R. A. (Ed.), CADE. Vol. 5663 of LNCS. Springer, pp. 140–145.
URL http://dx.doi.org/10.1007/978-3-642-02959-2_10

Wittocx, J., Denecker, M., Bruynooghe, M., Aug. 2013. Constraint propagation for first-order logic and inductive definitions. ACM Trans. Comput. Logic 14 (3), 17:1–17:45.
URL http://doi.acm.org/10.1145/2499937.2499938

You, J., Yuan, L., 1990. Three-valued formalization of logic programming: Is it needed? In: (PODS, 1990), pp. 172–182.
URL http://doi.acm.org/10.1145/298514.298559

Curriculum Vitae

Bart Bogaerts studied mathematics at the University of Leuven (KU Leuven), Belgium. His master thesis was titled "Unieke factorisatie in reguliere lokale ringen" (Unique factorisation in regular local rings) and was supervised by dr. Jan Schepers. He graduated summa cum laude in July 2011.

In September 2011, he joined the DTAI (Declaratieve Talen en Artificiële Intelligentie, Declarative Languages and Artificial Intelligence) research group of the Department of Computer Science of the KU Leuven to investigate knowledge representation and reasoning techniques, supervised by prof. dr. Marc Denecker. As his research interests evolved, prof. dr. Joost Vennekens and prof. dr. Jan Van den Bussche became his co-supervisors.

List of Publications

A complete and up-to-date list of publications can be found at http://www.cs.kuleuven.be/publicaties/lirias/mypubs.php?unum=U0078511.

Journal and Book Articles

- M. Denecker, B. Bogaerts, J. Vennekens. "The well-founded semantics is the principle of inductive definition: A study from first principles". Submitted to: Journal of the ACM.
- B. Bogaerts, J. Jansen, B. De Cat, G. Janssens, M. Bruynooghe and M. Denecker. "Bootstrapping inference in the IDP knowledge base system". Submitted to: New Generation Computing.
- B. Bogaerts, J. Vennekens and M. Denecker. "Grounded fixpoints and their applications in knowledge representation". Artificial Intelligence, volume 224, pages 51–71, 2015.
- B. De Cat, B. Bogaerts, M. Bruynooghe and M. Denecker. "Predicate logic as a modeling language: the IDP3 system". To be published in: M. Kifer and A. Liu, "Declarative Logic Programming: Theory, Systems, and Applications".
- M. Bruynooghe, H. Blockeel, B. Bogaerts, B. De Cat, S. De Pooter, J. Jansen, A. Labarre, J. Ramon, M. Denecker and S. Verwer. "Predicate logic as a modeling language: Modeling and solving some machine learning and data mining problems with IDP3". To be published in: Theory and Practice of Logic Programming.
- B. Bogaerts, B. De Cat, J. Jansen, M. Bruynooghe, B. De Cat, J. Vennekens and M. Denecker. "Simulating dynamic systems using linear

time calculus theories". In: Theory and Practice of Logic Programming, volume 14, issue 4–5, pages 477–492, 2014.

Peer-reviewed Articles at Conferences and Workshops

- B. Bogaerts, G. Van den Broeck. "Knowledge compilation of logic programs using approximation fixpoint theory". Submitted to: Thirty-first International Conference on Logic Programming, ICLP'15.
- B. Bogaerts, J. Vennekens and M. Denecker. "Partial grounded fixpoints". To be published in: Twenty-fourth International Conference on Artificial Intelligence, IJCAI'15.
- B. Bogaerts, J. Vennekens and M. Denecker. "Grounded fixpoints". In: Twenty-Ninth AAAI Conference on Artificial Intelligence, AAAI'15, Austin, January 25–29, 2015.
- B. Bogaerts, J. Jansen, B. De Cat, G. Janssens, M. Bruynooghe and M. Denecker. "Meta-level representations in the IDP knowledge base system: Towards bootstrapping inference engine development". In: International Workshop on Logic and Search, LaSh'14, Vienna, July 18, 2014.
- B. Bogaerts, J. Vennekens, M. Denecker and J. Van den Bussche. "Inference in the FO(C) modelling language". In: Twenty-first European Conference on Artificial Intelligence, ECAI'14, Prague, August 18–22, 2014.
- B. Bogaerts, J. Vennekens, M. Denecker and J. Van den Bussche. "Inference in the FO(C) modelling language". In: 14th International Workshop on Non-Monotonic Reasoning, NMR'14, Vienna, July 17–19, 2014.
- B. Bogaerts, J. Vennekens, M. Denecker and J. Van den Bussche. "FO(C) and related modelling paradigms". In: 14th International Workshop on Non-Monotonic Reasoning, NMR'14, Vienna, July 17–19, 2014.
- B. Bogaerts, J. Vennekens, M. Denecker and J. Van den Bussche. "FO(C): A knowledge representation language of causality". In: Technical Communications of the 30th International Conference on Logic Programming, ICLP'14, Vienna, July 19–22, 2014.

- J. Devriendt, B. Bogaerts and M. Bruynooghe. "BreakIDGlucose: On the importance of row symmetry in SAT". In: Fourth International Workshop on the Cross–Fertilization Between CSP and SAT, CSPSAT'14, Vienna, July 18, 2014.
- B. De Cat, B. Bogaerts, M. Denecker and J. Devriendt. "Model expansion in the presence of function symbols using constraint programming". In: 25th IEEE International Conference on Tools For Artificial Intelligence, ICTAI'13, Washington D.C., November 4–6, 2013.
- P. Van Hertum, J. Vennekens, B. Bogaerts, J. Devriendt and M. Denecker. "The effects of buying a new car: An extension of the IDP knowledge base system". In: Technical Communications of the 29th International Conference on Logic Programming, ICLP'13, Istanbul, August 24–29, 2013.
- T. Andrews, H. Blockeel, B. Bogaerts, M. Bruynooghe, M. Denecker, S. De Pooter, C. Macé and J. Ramon. "Analyzing manuscript traditions using constraint-based data mining". In: Proceedings of the First Workshop on Combining Constraint Solving with Mining and Learning, Montpellier, France, August 27, 2012
- H. Blockeel, B. Bogaerts, M. Bruynooghe, B. De Cat, S. De Pooter, M. Denecker, A. Labarre, J. Ramon, S. Verwer. "Modeling machine learning and data mining problems with FO($\cdot$)". In: Technical Communications of the 28th International Conference on Logic Programming, ICLP'12, Budapest, Hungary, September 4–8, 2012.
- J. Devriendt, B. Bogaerts, C. Mears, B. De Cat and M. Denecker. "Symmetry propagation: Improved dynamic symmetry breaking in SAT". In: 24th IEEE International Conference on Tools with Artificial Intelligence, ICTAI'12, Athens, Greece, November 7–9, 2012.

Peer-reviewed Abstracts

- J. Devriendt, B. Bogaerts, C. Mears, B. De Cat and M. Denecker. "Symmetry propagation: Improved dynamic symmetry breaking in SAT". In: SymCon, Quebec City, October 6–8, 2012.

Papers, Posters and Presentations at Miscellaneous Events

- B. De Cat, B. Bogaerts, M. Denecker. "MiniSAT(ID) for satisfiability checking and constraint solving". In: ALP Newsletter, September 2014
- B. De Cat, M. Denecker, P. Stuckey, B. Bogaerts. "Lazy model expansion: Interleaving grounding with search". Invited talk at International Workshop on Logic and Search (LaSh), 2014.
- B. Bogaerts. "Towards a knowledge base system for second order logics". Poster at 13th International Conference on Principles of Knowledge Representation and Reasoning (KR), 2012.

FACULTY OF ENGINEERING SCIENCE
DEPARTMENT OF COMPUTER SCIENCE
SCIENTIFIC COMPUTING GROUP
Celestijnenlaan 200A
B-3001 Heverlee
bart.bogaerts@cs.kuleuven.be
https://people.cs.kuleuven.be/bart.bogaerts