Abstract Dialectical Frameworks
An Analysis of Their Properties and Role in Knowledge Representation and Reasoning

Der Fakultät für Mathematik und Informatik
der Universität Leipzig
eingereichte

Habilitationsschrift

zur Erlangung des akademischen Grades

doctor rerum naturalium habilitatus
(Dr. rer. nat. habil.)

vorgelegt

von Dr. rer. nat. Hannes Straß

geboren am 12. Februar 1984 in Karl-Marx-Stadt (heute Chemnitz)

Die Annahme der Habilitationsschrift haben empfohlen:

1. Professor Dr. Gerhard Brewka, Universität Leipzig, Deutschland
2. Professor Dr. Marc Denecker, Katholieke Universiteit Leuven, Belgien
3. Professor Dr. Paul E. Dunne, Liverpool University, Vereinigtes Königreich

Beschluss über die Verleihung des akademischen Grades vom 23. Oktober 2017.

Preface

Acknowledgements . . . . . **v**
Abstract . . . . . **vii**
Kurzfassung . . . . . **ix**
Contents . . . . . **xix**

Acknowledgements

Many people have contributed in one way or another to the successful completion of this thesis. Whilst it is impossible to mention everyone by name, I would nonetheless like to express my gratitude towards the following.

Gerhard Brewka gave me the liberty to pursue my own research agenda and has also otherwise been tremendously supportive throughout my career.

My colleagues in Leipzig, Ringo Baumann, Stefan Ellmauthaler, Frank Loebe and Jörg Pührer always provided a pleasant working environment in which my ideas could thrive.

I consider myself happier for having collaborated with many people in the past years – often with enlightening discussions and fruitful results, so thank you all!

Over the years, I have also benefited from the criticism and suggestions of numerous anonymous reviewers. In addition, several people gave feedback on drafts of various documents: Bart Bogaerts pointed out that grounded models and F-stable models are the same; Sarah Alice Gaggl identified several missing links in Figure 6.1; Jörg Pührer made several useful suggestions for the improvement of Chapter 5; Johannes Peter Wallner suggested several examples for Chapter 3; Stefan Woltran provided a useful pointer to related work on realisability in logic programming.

My friends in Leipzig, Dresden and all over the place provided many welcome distractions and made everything worthwhile.

Finally, my family, especially my parents and my girlfriend, offered the constant support and encouragement that made all of this possible.

While I believe that all of those mentioned have contributed to an improved final version, none is, of course, responsible for remaining shortcomings.

Abstract

Abstract dialectical frameworks (ADFs) are a formalism for representing knowledge about abstract arguments and various logical relationships between them. This work studies ADFs in detail.

Firstly, we use the framework of approximation fixpoint theory to define various semantics that are known from related knowledge representation formalisms also for ADFs. We then analyse the computational complexity of a variety of reasoning problems related to ADFs. Afterwards, we also analyse the formal expressiveness in terms of realisable sets of interpretations and show how ADFs fare in comparison to other formalisms. Finally, we show how ADFs can be put to use in instantiated argumentation, where researchers try to assign meaning to sets of defeasible and strict rules.

The main outcomes of our work show that in particular the sublanguage of *bipolar* ADFs are a useful knowledge representation formalism with meaningful representational capabilities and acceptable computational properties.

Kurzfassung

Forschungsgegenstand der Künstlichen Intelligenz (KI) als Teilgebiet der Informatik ist die Automatisierung intelligenten Verhaltens. Seit seinen Anfängen in den Fünfzigerjahren des vorigen Jahrhunderts kann das Gebiet mittlerweile beträchtliche Fortschritte vorweisen. So wurden zum Beispiel auf dem Gebiet der Spiele die Menschen schrittweise bei immer mehr Spielen (bewiesenermaßen) chancenlos gegen entsprechend programmierte Maschinen. Die dahingehend womöglich erste von einer breiten Öffentlichkeit wahrgenommene solche Errungenschaft war wohl 1997 der Sieg von IBMs Schachcomputer „Deep Blue“ gegen den damaligen (menschlichen) Schachweltmeister Garri Kasparov. Seitdem hat die Entwicklung rasant zugenommen, so dass mittlerweile auch Dame[1] und Poker[2] (in der Variante „Heads-up/Limit/Hold'em“) als gelöst betrachtet werden können. Erst kürzlich konnte eine Forschungsgruppe unter der Leitung von David Silver ein Programm namens *AlphaGo* vorstellen, das das asiatische Brettspiel Go mindestens auf menschlichem Niveau beherrscht[3] und später auch deutlich gegen den amtierenden Go-Weltmeister Lee Sedol gewann, was noch vor zehn Jahren von vielen Fachleuten als „mehrere Jahrzehnte entfernt“ eingeschätzt wurde.

Neben der immer fortschreitenden Weiterentwicklung der reinen Rechenleistung von Computersystemen ist diese Entwicklung auch immer wieder auf neuartige Technologien der KI zurückzuführen. Im Fall von AlphaGo schreiben Silver et al. den Erfolg des Programms zu großen Teilen dem Einsatz so genannter tiefer künstlicher neuronaler Netze zu. Bei solchen neuronalen Netzen handelt es sich um vereinfachte Nachbildungen in der Natur vorgefundener neuraler Strukturen, die mit einer großen Anzahl von Übungsbeispielen angelernt werden und daraus eine implizite Bewertungsfunktion extrahieren, die (im Falle von Silver et al.) beim Spielen des Spiels günstige von ungünstigen (im Sinne einer gezielten Beeinflussung der Spielentwicklung hin zu einem eigenen Sieg) Zustände und Züge unterscheiden kann.

Auf Grund der Architektur künstlicher neuronaler Netze und der schieren Menge an von ihnen verarbeiteter Daten ist es jedoch für Menschen grundsätzlich nicht möglich, deren Funktionsweise im konkreten Einzelfall nachzuvollziehen. Während also Go-spielende *Menschen* erwartungsgemäß die Gründe für ihre Züge darlegen können, steht uns das Programm AlphaGo

[1] Jonathan Schaeffer, Neil Burch, Yngvi Björnsson, Akihiro Kishimoto, Martin Müller, Robert Lake, Paul Lu, and Steve Sutphen. Checkers is solved. *Science*, 317(5844):1518–1522, 2007. doi: 10.1126/science.1144079. URL `http://science.sciencemag.org/content/317/5844/1518`.

[2] Michael Bowling, Neil Burch, Michael Johanson, and Oskari Tammelin. Heads-up limit hold'em poker is solved. *Science*, 347(6218):145–149, 2015. doi: 10.1126/science.1259433. URL `http://science.sciencemag.org/content/347/6218/145`.

[3] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529:484–489, 2016. doi: 10.1038/nature16961. URL `http://dx.doi.org/10.1038/nature16961`.

gewissermaßen als Mysterium gegenüber, dessen Entscheidungen, wie es eine Kolumne im Fachmagazin *Nature* beschrieb, „hingenommen werden müssen“.[4]

Wir glauben jedoch, dass es für die menschliche Akzeptanz künstlich intelligenter Systeme hilfreich ist, wenn diese Systeme ihre Entscheidungen und Handlungen schlüssig begründen können. Darüber hinaus behaupten wir, dass es für eine produktive und vertrauensvolle Zusammenarbeit von Menschen und Maschinen wichtig ist, dass beide Seiten ihre Entscheidungen nachvollziehbar darlegen und in einem rationalen Diskurs erfolgreich verteidigen können.

Im Teilgebiet *Argumentationstheorie* der Künstlichen Intelligenz wird untersucht, wie sich die Analyse und der Austausch von Argumenten und Argumentationen rechnergestützt automatisieren lassen. Ein Argument wird dabei (z.B. nach Douglas Walton) definiert als eine Menge von Aussagen, von denen eine als Schlussfolgerung und die übrigen als Prämissen gelten, und zwischen Prämissen und Schlussfolgerung eine Ableitungsbeziehung besteht.[5] Walton führt weiterhin aus, dass ein Argument von anderen Argumenten unterstützt oder attackiert werden kann; Argumente können also nicht nur hinsichtlich ihrer intrinsischen Eigenschaften – wie z.B. der logischen Gültigkeit ihrer Ableitungsbeziehung – ausgewertet werden, sondern auch mittels ihrer sich ergebenden Interaktionen mit anderen Argumenten. Zur Verarbeitung von Argumenten wird natürlich zunächst eine Darstellung derselben benötigt; hierfür kommen zumeist Techniken der logikbasierten Wissensrepräsentation (KR; von engl. *Knowledge Representation*) zum Einsatz – Formalismen, durch die Wissen mit klarer Syntax und Semantik rechnergestützt darstellbar gemacht wird.

☞ *Abstrakte Argumentationsrahmenwerke und Abstrakte Dialektische Rahmenwerke*

Einer der am weitesten verbreiteten mathematischen Formalismen der Argumentationstheorie sind die 1995 von Phan Minh Dung vorgeschlagenen *Abstrakten Argumentationsrahmenwerke*.[6] Darin wird vollständig von intrinsischen Eigenschaften der analysierten Argumente abstrahiert, und diese als nicht weiter zerlegbare Einheiten behandelt. Einzig verfügbare Informationen über Argumente sind deren Beziehungen zu anderen Argumenten in Form einer Beziehung, die angibt, ob ein Argument ein anderes attackiert. Trotz – oder vielleicht gerade wegen – ihrer Einfachheit erfreuen sich diese *AFs* (von engl. *Argumentation Frameworks*) großer Beliebtheit in der Forschung, da sich darin viele grundlegende Fragen auf stark vereinfachte Weise formulieren und analysieren lassen.

Allerdings steht die Argumentationstheorie den Abstrakten Argumentationsrahmenwerken auch nicht auf einhellige Art uneingeschränkt positiv gegenüber: Verschiedene auf diesem Gebiet Forschende haben festgestellt, dass die mit AFs einhergehende Eingrenzung der Analyse *ausschließlich der Attacken* zwischen Argumenten die damit behandelbaren – in tatsächlichen Diskursen auftretenden – Phänomene nachhaltig einschränkt. Dieser Hauptkritikpunkt führte zu verschiedenen Ansätzen zur Verallgemeinerung von AFs, auf deren Einzelheiten wir jedoch nicht weiter eingehen und stattdessen auf einen einschlägigen Überblicksartikel verweisen.[7]

Einen der allgemeinsten Ansätze zur Verallgemeinerung von Dungs Abstrakten Argumentationsrahmenwerken stellen die von Brewka und Woltran 2010 vorgeschlagenen *Abstrakten*

[4] Unnamed authors. Digital intuition: A computer program that can outplay humans in the abstract game of Go will redefine our relationship with machines. *Nature*, 529:437, 2016. doi: 10.1038/529437a. URL `http://www.nature.com/news/digital-intuition-1.19230`. Editorial.

[5] Douglas Walton. Argumentation theory: A very short introduction. In *Argumentation in Artificial Intelligence*. Springer Dordrecht Heidelberg London New York, 2009.

[6] Phan Minh Dung. On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and *n*-Person Games. *Artificial Intelligence*, 77:321–358, 1995.

[7] Gerhard Brewka, Sylwia Polberg, and Stefan Woltran. Generalizations of Dung frameworks and their role in formal argumentation. *IEEE Intelligent Systems*, 29(1):30–38, 2014. ISSN 1541-1672. Special Issue on Representation and Reasoning.

Dialektischen Rahmenwerke (*ADFs*, von engl. *Abstract Dialectical Frameworks*) dar.[8] Darin wird ebenfalls von Argument-Interna abstrahiert; fundamentale Verallgemeinerung ist jedoch die, dass an Stelle einer reinen Angriffsbeziehung nun die Möglichkeit zur mathematischen Spezifikation *beliebiger* logischer Argumentbeziehungen tritt. So können beispielsweise durch (formalisierte Versionen von) Äußerungen wie „Argument *A* kann genau dann akzeptiert werden, wenn Argument *B* akzeptiert wird und Argument *C nicht* akzeptiert wird“ ausgedrückt werden, dass *A* von *B* unterstützt und von *C* angegriffen wird.

Die vorliegende Habilitationsschrift befasst sich umfassend mit Abstrakten Dialektischen Rahmenwerken. Wir definieren Semantiken für ADFs, analysieren die Berechnungskomplexität sich zu ihnen ergebender Entscheidungsprobleme, studieren ihre formale Ausdrucksstärke und Repräsentationseffizienz und präsentieren zuletzt noch eine beispielhafte Anwendung im Gebiet der anfechtbaren Regelsysteme. Nachfolgend stellen wir in deutscher Sprache kurz den Ausgangspunkt dieser Arbeit dar und geben ihre Hauptergebnisse wieder. Wir beginnen mit der Exposition einer Theorie über Operatoren in geordneten Strukturen, welche die Definition neuer Semantiken für Abstrakte Dialektische Rahmenwerke maßgeblich beeinflusst hat.

☜

Approximations-fixpunkttheorie

Ausgehend von einem fundamentalen Resultat von Knaster und Tarski über Fixpunkte von Operatoren in vollständigen Verbänden[9] entwickelten Denecker, Marek und Truszczyński einen algebraischen Ansatz zur Analyse formaler Semantiken mit Hilfe von Operatoren in geordneten Strukturen.[10] Der grundlegende Gedanke fußt dabei auf dem Konzept der *Approximation* eines Operators.

Aufgabe einer Semantik ist es, Elementen der Syntax (einer formalen Sprache) eine Menge zulässiger Interpretationen (ihre Modelle) zuzuordnen. Ein Ansatz zur Definition von Semantiken ist es, syntaktischen Elementen einen Operator auf Interpretationen so zuzuweisen, dass der Operator eingegebene Interpretationen im Sinne seines assoziierten syntaktischen Elements modifiziert. In dieser Leseweise stellt dann der kleinste Fixpunkt des Operators – jene kleinste Interpretation, die bei Eingabe in den Operator gleichermaßen als Ausgabe zurückgeliefert wird – das intendierte Modell des zum Operator gehörigen syntaktischen Elements dar, da intuitiv gesehen keine Modifikationen mehr notwendig bzw. möglich sind. Solche Ansätze zur Definition von Semantiken wurden bereits in den Siebzigerjahren des vorigen Jahrhunderts im Gebiet der Logikprogrammierung erfolgreich eingesetzt.

Für bestimmte Sprachen kann es jedoch der Fall sein, dass die für syntaktische Elemente definierten Operatoren nicht notwendigerweise die Eigenschaft der Monotonie erfüllen, welche gemäß dem Resultat von Knaster und Tarski hinreichend für die Existenz eines kleinsten Fixpunkts ist. Somit kann in solchen Fällen auch die Semantik solcher Sprachen nicht unmittelbar auf der Grundlage kleinster Fixpunkte definiert werden, da die Existenz solcher nicht garantiert werden kann. Denecker, Marek und Truszczyński begegnen diesem Problem mit Hilfe ihres Konzepts der Approximation von Operatoren. Arbeitet ein gegebener Operator auf einer Grundmenge von zweiwertigen Interpretationen, so gilt als Approximation dieses Operators jeder *solche* auf einer Grundmenge *dreiwertiger* Interpretationen arbeitende, der im Bezug auf zweiwertige Interpretationen ebenso abbildet wie der dadurch approximierte Operator, und zudem noch monoton bezüglich einer Ordnung der Informationserhaltung auf Interpretationen ist. Dreiwertige Interpretationen fügen dabei den klassischen Wahrheitswerten „wahr“

[8] Gerhard Brewka and Stefan Woltran. Abstract dialectical frameworks. In *Proceedings of the Twelfth International Conference on the Principles of Knowledge Representation and Reasoning (KR)*, pages 102–111, 2010.

[9] Alfred Tarski. A Lattice-Theoretical Fixpoint Theorem and Its Applications. *Pacific Journal of Mathematics*, 5(2): 285–309, 1955.

[10] Marc Denecker, Victor Marek, and Mirosław Truszczyński. Approximations, Stable Operators, Well-Founded Fixpoints and Applications in Nonmonotonic Reasoning. In *Logic-Based Artificial Intelligence*, pages 127–144. Kluwer Academic Publishers, 2000.

und „falsch“ noch einen dritten, nicht-designierten Wert „undefiniert“ (oder „unentschieden“, „unbekannt“) hinzu. Zwei dreiwertige Interpretationen stehen nun miteinander in Beziehung der Informationsordnung auf solchen, falls jedes Argument, dem in der einen Interpretation ein klassischer Wert zugewiesen wird, auch in der anderen Interpretation genau diesen Wert aufweist. Es lässt sich zeigen, dass der (informations-)kleinste Fixpunkt eines approximierenden Operators (welcher auf Grund der Monotonie existiert) die Fixpunkte des dadurch approximierten Operators in *dem* Sinne „approximiert“, dass letztere in einem Informationsintervall liegen, das von ersterem aufgespannt wird.

Mit Hilfe dieser und weiterer, darauf aufbauender Konzepte gelang es Denecker, Marek und Truszczyński in Folgearbeiten schließlich, wichtige Fragestellungen der logikbasierten Wissensrepräsentation umfassend zu beantworten.[11] Wir nutzen Approximationsfixpunkttheorie in dieser Arbeit, um die Semantik Abstrakter Dialektischer Rahmenwerke methodisch fundiert auszuarbeiten sowie sie mit Semantiken vergleichbarer Wissensrepräsentationsformalismen, wie Abstrakter Argumentationsrahmenwerke oder Logikprogramme, zu vergleichen.

☞ *Semantiken für Abstrakte Dialektische Rahmenwerke*

In dem Konferenzbeitrag, welcher Abstrakte Dialektische Rahmenwerke erstmals gegenüber der wissenschaftlichen Öffentlichkeit einführte, schlugen die Autoren Brewka und Woltran bereits einige Semantiken für ADFs vor. Unter anderem definierten sie eine Semantik der *zweiwertigen Modelle*, welche, wie hierin gezeigt, auch mit Hilfe eines Operators auf zweiwertigen Interpretationen definiert werden kann. Diesen Operator zugrundelegend verwenden wir anschließend Approximationsfixpunkttheorie, um weitere operator-basierte Semantiken für Abstrakte Dialektische Rahmenwerke zu definieren. Als Grundlage dafür dienen wiederum verschiedene mögliche Approximationen des ursprünglichen, zweiwertigen Operators. Je nach gewähltem Approximationsoperator führen existierende und hierin neu entwickelte Operatorsemantiken zu verschiedenen Familien von neuartigen Definitionen existierender Semantiken oder neuartigen Semantiken. Wir studieren hier (in Kapitel 3) im Detail die beiden Familien der *approximativen* Semantiken und der *ultimativen* Semantiken.

Der approximative Approximationsoperator ergibt sich aus der Übersetzung normaler Logikprogramme in ADFs (eine Übersetzung, welche von Brewka und Woltran vorgeschlagen wurde) und einer entsprechenden Verallgemeinerung eines dreiwertigen Operators, der zunächst bei Fitting[12] und später auch bei Denecker, Marek und Truszczyński zur Analyse und Rekonstruktion der Semantik der *stabilen Modelle* von Gelfond und Lifschitz[13] verwendet wurde. Der ultimative Approximationsoperator für Abstrakte Dialektische Rahmenwerke resultiert aus einer allgemeinen Definition von Denecker, Marek und Truszczyński, welche jedem zweiwertigen Operator denjenigen Approximator („ultimativer Approximator“ genannt) zuweist, welcher unter allen möglichen Approximatoren denjenigen mit dem für alle Eingaben höchsten Informationsgehalt auswählt.[14]

Legen wir nun einen bestimmten Approximationsoperator fest, so lässt sich die Menge der *vollständigen* Interpretationen (im Sinne der Approximationsfixpunkttheorie dreiwertige Fixpunkte) genau als die Menge der Fixpunkte dieses Operators definieren; darauf aufbauend folgt somit die *grundierte* Semantik als der kleinste Fixpunkt, die Menge der *zulässigen* Interpretationen als die Menge aller Postfixpunkte, die Menge der *bevorzugten* Interpretationen als

[11] Marc Denecker, V. Wiktor Marek, and Mirosław Truszczyński. Uniform Semantic Treatment of Default and Autoepistemic Logics. *Artificial Intelligence*, 143(1):79–122, 2003.

[12] Melvin Fitting. Fixpoint Semantics for Logic Programming: A Survey. *Theoretical Computer Science*, 278(1–2):25–51, 2002.

[13] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In *Proceedings of the International Conference on Logic Programming (ICLP)*, pages 1070–1080. The MIT Press, 1988.

[14] Marc Denecker, Victor W. Marek, and Mirosław Truszczyński. Ultimate approximation and its application in nonmonotonic knowledge representation systems. *Information and Computation*, 192(1):84–121, 2004.

die Menge der informationsmaximalen Postfixpunkte und schließlich die Menge der *semistabilen* Interpretationen als die Menge der reichweitenmaximalen zulässigen Interpretationen. (Die Reichweite einer dreiwertigen Interpretation meint hierbei genau diejenige Menge an Argumenten, welche von der Interpretation als akzeptiert/wahr oder abgelehnt/falsch festgesetzt wird, also genau solche, die nicht als unentschieden einem klassischen Akezptanzwert vorenthalten werden.) Für all diese Semantiken legen wir mit Hilfe eines geeigneten Operators für AFs dar, wie sie sich innerhalb der Approximationsfixpunkttheorie auf natürliche Weise aus den vorher bekannten (und gleichermaßen benannten) Semantiken der Abstrakten Argumentationsrahmenwerke ergeben.

Darüber hinaus zeigen wir, dass sich der ebenfalls von AFs bekannte – dort fundamentale – Begriff der *Konfliktfreiheit* auf verschiedene Weisen auf dreiwertige Interpretationen von ADFs verallgemeinern lässt. Wir bieten zwei mögliche operator-basierte Definitionen an: eine *asymmetrische* und eine *symmetrische* Variante der konfliktfreien Interpretationen. Für die symmterische Variante legen wir fest, dass ein Argument in einer dreiwertigen Interpretation als akzeptiert gelten kann, falls der gegebene Approximationsoperator in einem Revisionsschritt nicht festlegt, dass das Argument abzulehnen sei. Symmetrisch dazu (und daher der Name) legen wir fest, dass ein Argument in einer dreiwertigen Interpretation als abgelehnt gelten kann, falls der Operator in einem Revisionsschritt nicht festlegt, dass das Argument zu akzeptieren sei. Dies manifestiert eine weitere Absenkung der Akzeptanzstandards gegenüber der zulässigen Semantik: In einer zulässigen Interpretation ist es der Fall, dass akzeptierte Argumente vom Operator in einem Revisionsschritt auch tatsächlich als „zu akzeptieren“ bzw. abgelehnte Argumente auch tatsächlich als „abzulehnen“ klassifiziert werden. Für die asymmetrische Variante der dreiwertigen konfliktfreien Interpretationen stehen demnach sowohl symmetrisch konfliktfreie als auch zulässige Interpretationen Pate: Ein Argument kann hier schon als akzeptiert konstatiert werden, so nicht der Operator revidiert, dass es abzulehnen sei; andererseits kann ein Argument nur als abgelehnt konstatiert werden, falls der Operator revidiert, dass es tatsächlich abzulehnen sei. Aufbauend auf dem Begriff einer dreiwertigen Konfliktfreiheit (mit ihren verschiedenen Ausprägungen) lassen sich nun auch maximierungsbasierte Kriterien (wie von der bevorzugten Semantik bekannt) auf dieses Konzept anwenden. Dieses führt zur Semantik der *naiven* Interpretationen, nämlich der Menge aller konfliktreien Interpretationen, die im Bezug auf ihre Konfliktfreiheit informationsmaximal sind, und zur Semantik der *stufigen* Interpretationen, der Menge der reichweitenmaximalen konfliktfreien Interpretationen. Die Wahl eines Operators (approximativ/ultimativ) und die Wahl einer Konfliktfreiheit sind dabei orthogonal zueinander, so dass jeweils mindestens vier naive und stufige Semantiken existieren, zum Beispiel die approximativen symmetrisch konfliktfreien naiven Interpretationen oder die ultimativen asymmetrisch konfliktfreien stufigen Interpretationen.

Da sich die von uns quasi in Form von Korollaren definierten Semantiken problemlos auf abstrakt gehaltene Ansätze von beliebigen Approximationsoperatoren in vollständigen partiellen Ordnungen verallgemeinern lassen, bieten wir für solche Ansätze noch zwei Resultate an, die die Existenz bevorzugter und naiver Interpretationen (bzw. Verallgemeinerungen davon) in nur sehr schwach eingeschränkten Verhältnissen garantieren.

Zuätzlich zur Herausarbeitung der Verbindungen der ADF-Semantiken zu denen der Abstrakten Argumentationsrahmenwerke erzielen wir auch vielerlei Ergebnisse um beide genannten Formalismen zu den normalen aussagenlogischen Logikprogrammen ihrerseits in Beziehung zu setzen. So betrachten wir zwei in der einschlägigen Literatur vorgefundene Übersetzungen von AFs in Logikprogramme und die dafür geltenden Koinzidenz- bzw. Äquivalenzbeziehungen zwischen jenen Semantiken, die sowohl für AFs als auch für normale aussagenlogische Logikprogramme definiert sind. Gleichermaßen analysieren wir ebensolche Beziehungen zwischen normalen Logikprogrammen und Abstrakten Dialektischen Rahmen-

☞ werken.

Bipolare Abstrakte Dialektische Rahmenwerke

Bereits in ihrer Arbeit von 2010, die gewissermaßen die Grundsteinlegung aller weiterer Forschung an Abstrakten Dialektischen Rahmenwerken darstellte, beobachteten Brewka und Woltran, dass bei der Verwendung von ADFs nicht in jedem Falle die gesamte Ausdrucksstärke der Akzeptanzbedingungen von Nöten ist. Ihre Erkenntnis gossen Brewka und Woltran in eine fundamentale Definition, die mittlerweile auf ein breit gefächertes Spektrum an von ihr beeinflussten Begriffen blicken kann. Demnach heißt ein Abstraktes Dialektisches Rahmenwerk *bipolar*, falls alle Abhängigkeitsbeziehungen zwischen Argumenten gemäß deren Akzeptanzbedingungen zweifelsfrei in mindestens eine von zwei Kategorien eingeordnet werden können, nämlich unterstützende und angreifende Abhängigkeitsbeziehungen. Eine Beziehung von einem Argument A zu einem Argument B heißt nach jener Fassung *unterstützend*, falls es keine Interpretation der Elternargumente von B (also derjenigen Argumente, von deren Akzeptanz oder Nicht-Akzeptanz die Akzeptanz von B letztendlich abhängt) so gibt, dass B gemäß seiner Akzeptanzbedingung und der vorherrschenden Konfiguration akzeptiert wird, jedoch die Änderung der Position im Bezug auf A gemäß der Akzeptanzbedingung von B die Auswirkung hätte, dass B nicht mehr akzeptiert wird – gewissermaßen die Änderung in der Akzeptanz von A alleinig als die Ursache der späteren Nicht-Akzeptanz von B gesehen werden muss. Dazu spiegelgleich heißt die Beziehung von A nach B *angreifend*, falls es keine Interpretation so gibt, dass B zunächst nicht akzeptiert wird, aber nach ausschließlicher Modifikation der Akzeptanz von A (von „abgelehnt“ zu „akzeptiert“) und ansonsten gleichbleibender Gesamtkonfiguration auch B akzeptiert wird.

Die sich dadurch ergebende Klasse von Abstrakten Dialektischen Rahmenwerken stellt eine echte Einschränkung der allgemeinen ADFs dar, da in solchen auch gewissermaßen kontextabhängig unterstützend oder angreifend agierende Beziehungen zwischen Argumenten möglich sind, wie zum Beispiel in der Akzeptanzbedingung „das Argument wird genau dann akzeptiert, wenn entweder beide Eltern akzeptiert sind oder beide Eltern abgelehnt sind“. Nichtsdestotrotz genügten die verbleibenden Ausdrucksmittel der bipolaren ADFs an anderer Stelle Brewka und Gordon, die noch im selben Jahr (2010) aufzeigten, wie das System „Carneades“, das zur Modellierung konkreter argumentativer Strukturen dient, mit Hilfe von Abstrakten Dialektischen Rahmenwerken *so* rekonstruiert werden konnte, dass die Semantik nun auch in der Lage war, mit *zyklischen* Abhängigkeiten zwischen Aussagen umzugehen, was im ursprünglichen Carneades wegen technischer Einschränkungen nicht möglich war.[15] Wollen wir auch nicht spätere Entwicklungen vorweg nehmen, so gebietet es doch die Vollständigkeit, darauf hinzuweisen, dass unsere eigene Arbeit (hierin als Kapitel 6 einbezogen) zur Anwendung von Abstrakten Dialektischen Rahmenwerken auf dem Gebiet der anfechtbaren Regelsysteme ebenso deutlich die Einsatzfähigkeit bipolarer ADFs vor Augen führt. Umso deutlicher wird dies mit der erfolgten Analyse der Berechnungskomplexität mit Abstrakten Dialektischen Rahmenwerken assoziierter Entscheidungs- und Funktionsprobleme.

☞

Berechnungskomplexität

Die Theorie der Berechenbarkeit mathematisch gefasster Funktionen liefert uns ein allgemein anerkanntes, probates Mittel zur Analyse der fundamentalen Fragestellung, ob ein gegebenes Problem mit Hilfe von rechnenden Maschinen gelöst werden kann. Darauf aufbauend untersucht die *Komplexitätstheorie*, welche zur Berechnung nötigen Ressourcen in welchem qualitativen Umfang zur umfassenden Lösung einer rechnerlösbaren Fragestellung vorgehalten werden müssen. In der Literatur haben sich dabei als fundamentale Ressourcen die *Zeit* im Sinne einer Dauer der Berechnung, gemessen in einzelnen Rechenschritten, und der

[15] Gerhard Brewka and Thomas F. Gordon. Carneades and abstract dialectical frameworks: A reconstruction. In *Proceedings of the Third International Conference on Computational Models of Argument (COMMA)*, volume 216 of *FAIA*, pages 3–12. IOS Press, September 2010.

Platz im Sinne des Speicherverbrauchs einer Berechnung, gemessen in einzelnen Speicherzellen zur Ablage jeweils eines Zeichens eines vorher vereinbarten Alphabets, herauskristallisiert. Grundlegendes Alleinstellungsmerkmal der Komplexitätstheorie ist nun jedoch nicht die feingliedrige Analyse der Schritte einzelner Berechnungen, sondern vielmehr das Vermögen, sich von konkreten Lösungsansätzen für gegebene Probleme entbinden und an Stelle dessen die inhärente Komplexität *des Problems selbst* untersuchen zu können.[16]

Solche Untersuchungen geschehen häufig mit Hilfe so genannter *Komplexitätsklassen* – Mengen von Problemen, welche allesamt den gleichen Ressourcenverbrauch aufweisen. „Probleme" bezieht sich hier meist auf *Entscheidungsprobleme*, bei denen die Aufgabe darin besteht, eine gegebene (ein Objekt aus dem Anschauungsbereich des Problems kodierende) Zeichenkette als ein Wort (*positive* Instanz) oder Nicht-Wort (*negative* Instanz) einer bestimmten – somit das Problem charakterisierenden – formalen Sprache (über einem vereinbarten Alphabet) zu klassifizieren. In dieser Form könnte zum Beispiel das Problem der Erkennung von Primzahlen solchermaßen gefasst werden, dass die charakterisierende formale Sprache über einem Alphabet, das die Zeichen 0 und 1 umfasst, genau diejenigen Zeichenketten enthält, deren Interpretation als binär kodierte natürliche Zahlen eine Primzahl ergeben. Mittlerweile ist bekannt, dass sich dieses Problem in der Komplexitätsklasse P befindet, demzufolge sich die maximale Anzahl der Rechenschritte in Abhängigkeit der Länge der eingegebenen Zeichenkette durch ein konkretes Polynom „von oben" abschätzen lässt.[17]

Eine weitere, in unserer hierin (speziell in Kapitel 4) vorgestellten Untersuchung der Berechnungskomplexität der Abstrakten Dialektischen Rahmenwerke häufig vorkommende Komplexitätsklasse ist NP, deren Zugehörige genau diejenigen Probleme sind, für die sich positive Instanzen in P als solche verifizieren lassen. Auf Grundlage dieser beider Klassen lässt sich noch eine Vielzahl weiterer solcher Zusammenfassungen von Problemstellungen definieren: Die Klasse coNP enthält danach genau die Komplementärprobleme der Mitglieder von NP; die Klasse Σ_2^P enthält genau diejenigen Probleme, deren positive Instanzen sich in P mit Hilfe eines „NP-Orakels" verifizieren lassen, die Klasse Π_2^P genau die, deren negative Instanzen sich in P mit Hilfe eines NP-Orakels verifizieren lassen. Ein NP-Orakel ist dabei ein gedankliches Konstrukt, das intuitiv als eine Subroutine gesehen werden kann, die jede beliebige Fragestellung aus NP in einem einzigen Rechenschritt entscheiden kann. Die Definitionen der Begriffe Σ_2^P und Π_2^P lassen sich von 2 auf beliebige natürliche Zahlen erweitern, wodurch die so genannte *Polynomielle Hierarchie* zustande kommt, eine unendliche Abfolge ineinander enthaltener Komplexitätsklassen, deren „obere Schranke", wenn man so will, nur festlegt, dass alle darin enthaltenen Problemstellungen mit höchstens polynomiellem Platzverbrauch entscheidbar sind.

Bei unserer Analyse der Berechnungskomplexität der Abstrakten Dialektischen Rahmenwerke können wir herausarbeiten, dass die Kapazitäten zur Problemmodellierung ebendieser teilweise merklich über die der Abstrakten Argumentationsrahmenwerke hinausgeht. Konkret untersuchen wir vier verschiedene Problemtypen, welche sich aus dem Umgang mit ADFs ergeben: 1. Gegeben ein ADF, eine Semantik und eine Interpretation, erfüllt die Interpretation die Maßgaben der Semantik? 2. Gegeben ein ADF und eine Semantik, besitzt das ADF eine nicht-triviale Interpretation unter dieser Semantik? (Eine Interpretation gilt als nicht-trivial, wenn in ihr zumindest ein Argument akzeptiert oder abgelehnt ist.) 3./4. Gegeben ein ADF, eine Semantik und ein Argument, wird dieses Argument in einer/jeder Interpretation der Semantik akzeptiert? Letztere beiden Probleme sind auch unter den Namen *leichtgläubiges*

[16] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.

[17] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. PRIMES is in P. *Annals of Mathematics*, 160(2):781–793, 2004. doi: 10.4007/annals.2004.160.781.

und *skeptisches* Schließen bekannt, da jeweils nach logischen Konsequenzen des gegebenen Abstrakten Dialektischen Rahmenwerks gefragt wird. Ohne zu sehr ins Detail zu gehen, lässt sich sagen, dass die Berechnungskomplexität der meisten dieser Probleme für die untersuchten Semantiken im Falle von Abstrakten Dialektischen Rahmenwerken in der Polynomiellen Hierarchie üblicherweise eine Ebene über der jeweiligen Komplexität desselben Problems für Abstrakte Argumentationsrahmenwerke liegt.

Als abschließendes Hauptergebnis der hier (in Kapitel 4) vorgestellten Darlegung lässt sich zweifellos konstatieren, dass sich für die Teilklasse der bipolaren Abstrakten Dialektischen Rahmenwerke die Berechnungskomplexität der oben genannten Entscheidungsprobleme in den allermeisten Fällen im Vergleich zu Abstrakten Argumentationsrahmenwerken *nicht* ändert. Anders ausgedrückt bieten bipolare ADFs im Gegensatz zu AFs die zusätzlichen Ausdrucksmittel der unterstützenden Beziehungen zwischen Argumenten, ohne dass diese Ausdrucksmittel, salopp formuliert, in Form von erhöhtem Ressourcenaufwand „bezahlt" werden müssen.

☞ *Aussdrucksstärke und Repräsentationseffizienz*

Während die erhöhte „Ausdrucksstärke" von ADFs durch die Verfügbarkeit von unterstützenden Argumentbeziehungen intuitiv klar zu sein scheint, so ist doch eine genauere Fassung des Begriffs und eine damit einhergehende umfassendere Darlegung seiner Ausprägungen im Falle konkreter Argumentationsformalismen erhellend. In einem mittlerweile klassischen Beitrag zur logikbasierten Wissensrepräsentation schlugen Gogic, Kautz, Papadimitriou und Selman im Jahr 1995 Begriffe vor, mit deren Hilfe sich die intuitiven Konzepte *Ausdrucksstärke* und *Repräsentationseffizienz* auf präzise Weise mathematisch fassen lassen.[18] Nach dieser Konzeption werden logikbasierte Wissensrepräsentationsformalismen als alternative Möglichkeiten zur Darstellung formaler Sprachen (wie in der Berechnungs- und Komplexitätstheorie zum Beispiel durch Automatenmodelle üblich) betrachtet, und ein Formalismus $\mathcal{F}_2$ gilt als *mindestens so ausdrucksstark* wie ein Formalismus $\mathcal{F}_1$, wenn sich alle Interpretationsmengen, die sich in $\mathcal{F}_1$ ausdrücken lassen (mittels einer Semantik und den in $\mathcal{F}_1$ verfügbaren Mitteln) auch innerhalb von $\mathcal{F}_2$ ausdrücken lassen. Dieses Konzept der *relativen Ausdrucksstärke* interessiert sich zunächst also nur für die bloße Fähigkeit, eine Modellmenge (Sprache) ausdrücken zu können, was jedoch seinerseits die Grundlage dafür legt, in späteren Weiterentwicklungen auch die *Repräsentationseffizienz* von Formalismen mit formalen Methoden zu fassen. Ebenfalls nach einem Begriffsvorschlag von Gogic et al. gilt ein Formalismus $\mathcal{F}_2$ als *mindestens so repräsentationeffizient* (auch: mindestens so *sukzinkt*) wie ein Formalismus $\mathcal{F}_1$, wenn für jeden Ausdruck von $\mathcal{F}_1$, dessen Interpretationsmenge sich sowohl in $\mathcal{F}_1$ als auch $\mathcal{F}_2$ ausdrücken lässt, ein Ausdruck von $\mathcal{F}_2$ existiert, dessen Größe polynomiell durch die Größe des Ausdrucks von $\mathcal{F}_1$ beschränkt ist; „polynomiell" erfordert dabei die Existenz *eines festen* Polynoms, das als obere Schranke für sämtliche zu betrachtenden Ausdrücke dient. Anders ausgedrückt ist $\mathcal{F}_2$ exponentiell repräsentationseffizienter (sukzinkter) als $\mathcal{F}_1$, falls $\mathcal{F}_2$ mit höchstens polynomieller Aufblähung alles ausdrücken kann, was $\mathcal{F}_1$ ausdrücken kann, es aber einen Ausdruck in $\mathcal{F}_2$ gibt, dessen kleinstmöglicher äquivalenter Ausdruck in $\mathcal{F}_1$ exponentiell größer ist.

☞ *Aussdrucksstärke und Repräsentationseffizienz Abstrakter Dialektischer Rahmenwerke*

Unter Verwendung der oben definierten Begriffe zur Fassung relativer Ausdrucksstärke und Repräsentationseffizienz studieren wir (in Kapitel 5) die entsprechenden Beziehungen zwischen Abstrakten Argumentationsrahmenwerken, Abstrakten Dialektischen Rahmenwerken, der klassischen Aussagenlogik und verschiedenen Ausprägungen aussagenlogischer Logikprogramme. Dabei können wir herausstellen, dass die genannten Formalismen üblicherweise eine

[18] Goran Gogic, Henry Kautz, Christos Papadimitriou, and Bart Selman. The comparative linguistics of knowledge representation. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 862–869. Morgan Kaufmann, 1995.

strenge Hierarchie bezüglich ihrer Ausdrucksstärke bilden, wobei die Abstrakten Argumentationsrahmenwerke jeweils die am wenigsten ausdrucksstarken Glieder darstellen, von den echt ausdrucksstärkeren bipolaren Abstrakten Dialektischen Rahmenwerken gefolgt werden, und beide ihrerseits durch die (unter der Semantik der zweiwertigen Modelle) universell ausdrucksstarken uneingeschränkten ADFs und die Aussagenlogik dominiert werden. Aussagenlogische Logikprogramme liegen dabei je nach gewählter Semantik (unterstützte oder stabile Modelle) und gewählter Teilklasse solcher (normale oder kanonische Logikprogramme) entweder zwischen bipolaren ADFs und AFs oder obenauf mit Aussagenlogik und allgemeinen ADFs. Im Hinblick auf die Korrespondenz der Berechnungskomplexitäten zwischen Abstrakten Argumentationsrahmenwerken und bipolaren Abstrakten Dialektischen Rahmenwerken kann die formell nachgewiesene höhere Ausdrucksstärke der bipolaren ADFs also uneingeschränkt als eines der wichtigsten Ergebnisse dieser Untersuchung gelten.

Hinsichtlich der Repräsentationseffizienz der betrachteten Formalismen ergibt sich im Wesentlichen, dass bipolare Abstrakte Dialektische Rahmenwerke unter der Semantik der zweiwertigen Modelle exponentiell sukzinkter sind als normale Logikprogramme unter der Semantik der unterstützten Modelle; im Übrigen sind allgemeine (uneingeschränkte) Abstrakte Dialektische Rahmenwerke unter der Semantik der zweiwertigen Modelle im Vergleich zur klassischen Aussagenlogik (unter ihrer Standardsemantik) in ihrer Repräsentationseffizienz als gleichwertig zu betrachten.

Anwendung Abstrakter Dialektischer Rahmenwerke auf anfechtbare Regelsysteme

Im Forschungsgebiet der abstrakten formalen Argumentation gibt es ein Teilgebiet, das sich mit der Nutzung von Techniken der abstrakten Argumentation zur Definition von Semantiken für anfechtbare Regelsysteme befasst. Anfechtbare Regelsysteme sind ein einfacher, logik-inspirierter Wissensrepräsentationsformalismus, bei dem über einem endlichen Vokabular aus aussagenlogischen Literalen, also Atomen und Negationen solcher, direktionale Regeln zweierlei Art formuliert werden können, nämlich einerseits *strikte* Regeln und andererseits *anfechtbare* Regeln. Jedwede Regeln bestehen aus einem *Kopf* – einem Literal –, welcher die Konklusion eines durch die Regeln explizierten logischen Schlusses darstellt, und einem *Körper* – einer Folge von Literalen –, der wiederum die Prämissen ebendieses Schlusses denotiert. Bei jeder Ansammlung von strikten und anfechtbaren Regeln spricht man von einem *anfechtbaren Regelsystem*. Die dem zugrundeliegende Eingebung legt fest, dass strikte Regeln eine universell geltende Schlussbeziehung zwischen Körper und Kopf konstatieren, während ebendiese Schlussbeziehung im Falle anfechtbare Regeln gerade anfechtbar ist, also von einer etwaigen Bedeutungslehre im Falle widersprüchlicher Informationen oder aus anderen übergeordneten Gründen außer Kraft gesetzt werden kann. Demzufolge stellen die anfechtbaren Regelsysteme eine Form des Ermangelungsschließens dar, da die mit Hilfe anfechtbarer Regeln getroffenen Schlussfolgerungen ihrem Wesen nach „in Ermangelung vollständiger Informationen“ stattfanden und ihre Rücknahme jederzeit durch die Hinzunahme anderslautender Aussagen erforderlich gemacht werden kann.

Zur argumentationsbasierten Analyse der Bedeutung solcher anfechtbaren Regelsysteme hat die Fachliteratur im Großen und Ganzen zwei verschiedene Ansätze zu bieten. Einen ersten Vorschlag unterbreiteten Caminada und Amgoud, die angaben, wie aus Regeln induktiv „Argumente“ konstruiert und nach welcher Maßgabe zwischen den so konstruierten Objekten Angriffsbeziehungen konstituiert werden können.[19] Demnach haben „Argumente“ eine ineinander verschachtelte Struktur, die grundsätzlich ganze Argumentationsfolgen von Fakten und Annahmen ausgehend hin zu Schlussfolgerungen abbildet. Diese Autoren stellten in derselben Arbeit jedoch schlussendlich fest, dass eine solche Herangehensweise nur mit zusätzlichen

[19] Martin Caminada and Leila Amgoud. On the evaluation of argumentation formalisms. *Artificial Intelligence*, 171 (5–6):286–310, 2007.

Einschränkungen zu befriedigenden Ergebnissen führt: So müssen verwendete Semantik und Definition der Angriffsbeziehungen sauber aufeinander abgestimmt werden, um das Auftreten aus menschlich-alltäglicher Sicht unerwarteter Schlüsse ausschließen zu können.

In einer publizistischen Replik dazu schlugen Wyner und Kollegen einen anders gearteten Ansatz vor, bei dem die im entstehenden Argumentationsrahmenwerk definierten „Argumente“ keine innere Struktur besitzen, sondern sich gemeinhin auf der Ebene der Literale und Regeln der Ausgangssprache bewegen, und argumentative Konstrukte vielmehr aus den Angriffsbeziehungen zwischen verschiedenen Argumentknoten im abstrakten Rahmenwerk gewissermaßen auf einer übergeordneten Ebene hervorgehen.[20] Wyner und Kollegen begründeten diese andersartige Ausrichtung ihres Ansatzes mit philosophischen Betrachtungen über den *Argumentbegriff* an sich, auf die wir hier nicht näher eingehen wollen. Jedoch ist der Ansatz von Wyner et al. in technischer Hinsicht vorteilhaft gegenüber dem von Caminada und Amgoud, da bei letzterem die Anzahl der entstehenden „Argumente“ gemeinhin nicht beschränkt ist, also selbst aus einem gegebenen endlichen anfechtbaren Regelsystem ein unendlich großes Abstraktes Argumentationsrahmenwerk entstehen kann, wohingegen bei Wyner et al. die Größe des sich ergebenden AFs mittels der Größe des eingegebenen anfechtbaren Regelsystems abgeschätzt werden kann. Jedoch lässt aus unserer Sicht der in seiner Grundkonzeption bereits fortgeschrittene Ansatz von Wyner und Kollegen immer noch Wünsche offen, zum Beispiel die vollständig nachgewiesene Abwesenheit kontraintuitiver oder irrationaler Schlussweisen, so dass wir (in Kapitel 6) einen Ansatz präsentieren, der anfechtbare Regelsysteme in Abstrakte Dialektische Rahmenwerke übersetzt und ersteren dadurch vermöge der ADF-Semantiken eine Bedeutung zuweist. Wir können dazu unter anderem zeigen, dass diese Übersetzung nicht nur effizient konstruierbar ist, sondern auch irrationale Folgerungen (gemäß einer Begriffsfassung von Caminada und Amgoud) zuverlässig ausschließt.

☞ *Zusammenfassung, weitere Arbeiten und Ausblick*

Die hier vorliegende Monographie stellt im Bezug auf den Umfang der darin detailliert behandelten Fragestellungen einen wichtigen Beitrag zur wissenschaftlichen Arbeit an Abstrakten Dialektischen Rahmenwerken dar. Beginnend mit den semantischen Grundlagen und Zusammenhängen zu anderen Formalismen analysieren wir auch Berechnungskomplexität und Ausdrucksstärke sowie Repräsentationseffizienz, um schließlich mit einer beispielhaften Anwendung von ADFs zu schließen.

Es liegt jedoch auch in der Natur dieses Dokumentes als Qualifikationsarbeit, dass wir auf mehrere Arbeiten über Abstrakte Dialektische Rahmenwerke nicht im gleichen Detailgrad eingehen, da sie in Zusammenarbeit mit oder ausschließlich von anderen Forschenden erzielt wurden. Auf eine genauere Auflistung möchten wir hier verzichten und verweisen auf die englischsprachige Variante einer solchen in Kapitel 7.

Können wir auch mit den hierin vorgestellten Ergebnissen maßgeblich zu einem verbesserten Verständnis der Abstrakten Dialektischen Rahmenwerke beitragen, so verbleiben trotz allem naturgemäß eine Reihe von Fragestellungen, die wir schlichtweg zukünftigen Arbeiten zurechnen müssen. Es sei zum Beispiel erwähnt, dass für AFs weitere Semantiken existieren, die unseres Wissens nach noch nicht auf ADFs verallgemeinert wurden, etwa die Semantik der *sehnlichen* Interpretationen, welche für AFs strukturähnlich zur Semantik der *idealen* Interpretationen definiert werden kann. Viele andere Aspekte, die für Abstrakte Argumentationsrahmenwerke bereits gut erforscht sind, stellen für die Abstrakten Dialektischen Rahmenwerke noch Neuland dar, so zum Beispiel das Verhalten der Semantiken unter Einbezug dynamischer Aspekte, welche sich am ohnehin inhärent dynamischen realweltlichen Argumentieren orientieren und darauf potenziell wieder zurückwirken könnten.

[20] Adam Wyner, Trevor Bench-Capon, and Paul Dunne. Instantiating knowledge bases in abstract argumentation frameworks. In *Proceedings of the AAAI Fall Symposium – The Uses of Computational Argumentation*, 2009.

Contents

1 Introduction **1**
1.1 Publications . . . 3

2 Background **7**
2.1 Mathematical Notation . . . 7
2.2 Logic in Knowledge Representation and Reasoning . . . 7
2.3 Lattice Theory . . . 9
2.4 Approximation Fixpoint Theory . . . 10
2.4.1 Logic Programming . . . 12
2.5 Abstract Argumentation . . . 13
2.5.1 Abstract Argumentation Frameworks . . . 13
2.5.2 Abstract Dialectical Frameworks . . . 14
2.6 Complexity Theory . . . 17

3 Defining Semantics via Approximation Fixpoint Theory **19**
3.1 Approximate Semantics of ADFs . . . 24
3.1.1 The Characteristic Approximate Operator of an ADF . . . 24
3.2 Relationship to Normal Logic Programs . . . 33
3.2.1 From ADFs to Logic Programs . . . 33
3.2.2 From Logic Programs to ADFs . . . 35
3.3 Ultimate Semantics of Abstract Dialectical Frameworks . . . 37
3.4 AF Semantics as Special Cases . . . 40
3.4.1 Fixpoint Semantics for Abstract Argumentation Frameworks . . . 41
3.4.2 From Argumentation Frameworks to Logic Programs . . . 44
3.4.3 From Logic Programs to Argumentation Frameworks . . . 48
3.5 General Semantics for Approximating Operators . . . 48
3.5.1 Admissible . . . 48
3.5.2 Semi-stable . . . 49
3.5.3 Conflict-free (Asymmetric) . . . 49
3.5.4 Conflict-free (Symmetric) . . . 51
3.5.5 Naive . . . 52
3.5.6 Stage . . . 52
3.6 Existence Results for General Operators . . . 53
3.7 Overview of Results . . . 55
3.8 Concluding Remarks . . . 56

4 Computational Complexity **59**
4.1 Preparatory Considerations . . . 61
4.1.1 Notation and Decision Problems . . . 61
4.1.2 Relationship Between the Operators . . . 63
4.1.3 Reductions and Encoding Techniques . . . 65
4.1.4 Operator Complexities . . . 68
4.1.5 Generic Upper Bounds . . . 70
4.2 General ADFs . . . 72
4.2.1 Symmetric Conflict-free Semantics . . . 72
4.2.2 Admissibility-based Semantics . . . 78
4.2.3 Two-valued Semantics . . . 86
4.2.4 Overview . . . 90
4.3 Bipolar ADFs . . . 92
4.3.1 Symmetric Conflict-free Semantics . . . 94
4.3.2 Two-valued Semantics . . . 95
4.3.3 Overview . . . 96
4.4 Conclusion . . . 98

5 Relative Expressiveness and Succinctness **99**
5.1 Background on Relative Expressiveness . . . 103
5.1.1 Translations Between Considered Formalisms . . . 104
5.1.2 Representing Bipolar Boolean Functions . . . 106
5.2 Relative Expressiveness . . . 108
5.2.1 Supported Semantics . . . 108
5.2.2 Stable Semantics . . . 121
5.2.3 Supported vs. Stable Semantics . . . 122
5.3 Allowing Vocabulary Expansion . . . 123
5.4 Discussion . . . 127

6 An Application to Theory Bases **129**
6.1 Background on Defeasible Theories . . . 130
6.2 Instantiations to Abstract Argumentation Frameworks . . . 131
6.2.1 The Approach of Caminada and Amgoud (2007) . . . 131
6.2.2 The Approach of Wyner, Bench-Capon, and Dunne (2013) . . . 133
6.3 Instantiations to Abstract Dialectical Frameworks . . . 135
6.3.1 From Theory Bases to ADFs . . . 135
6.3.2 Support Cycles in Theory Bases . . . 138
6.3.3 Inconsistent Theory Bases . . . 140
6.3.4 Properties of the Translation . . . 140
6.4 A Direct Semantics for Defeasible Theory Bases . . . 142
6.4.1 Relationship to Autoepistemic Logic . . . 147
6.4.2 Defining Further Semantics . . . 148
6.5 Conclusion . . . 149

7 Discussion **151**
7.1 Related and Possible Future Work . . . 151

Bibliography **152**

Index **163**

Chapter 1

Introduction

The goal of artificial intelligence (AI) is to create intelligent machines. These artificial entities, be they purely software or physically embedded, should be able to think and act rationally in order to fulfil goals. The possession of knowledge is one of the most important aspects of intelligence, as it seems to be a culture of acquiring and passing on knowledge that has enabled human primates to develop from bands of hunters and gatherers to technological nations.

Knowledge representation (KR) is a sub-field of AI that investigates formal methods for representing knowledge and reasoning with such representations. One motivation for doing so is that to exhibit intelligence, it is not enough to store and recall facts, but moreover facts should be linked together to infer new facts, and the knowledge altogether should be accessible such that it enables purposeful action. In KR, typical representations of knowledge are *explicit* and *symbolic*, which means that formal symbols are used to refer to domain entities (things the representation talks about), and are manipulated to emulate reasoning. To ensure that this reasoning is sound and complete, KR often makes use of techniques from the field of formal logic.

On the other hand, alternative approaches to artificial intelligence have made some impressive advances in the past years. For example, in the domain of playing games, computers are ever more adept at games that were previously thought to be "too hard for machines", such as Checkers (Schaeffer, Burch, Björnsson, Kishimoto, Müller, Lake, Lu, and Sutphen, 2007), Poker (Bowling, Burch, Johanson, and Tammelin, 2015) and most recently Go (Silver, Huang, Maddison, Guez, Sifre, van den Driessche, Schrittwieser, Antonoglou, Panneershelvam, Lanctot, Dieleman, Grewe, Nham, Kalchbrenner, Sutskever, Lillicrap, Leach, Kavukcuoglu, Graepel, and Hassabis, 2016). Especially for the last work, the authors attribute their success in part to the use of *deep artificial neural networks*, a biology-inspired approach where "knowledge" (if at all) exists only implicitly and on a sub-symbolic level. A Nature editorial on the work of Silver et al. (2016) comments that "AlphaGo cannot explain how it chooses its moves [...] As shown by its results, the moves that AlphaGo selects are invariably correct. But the interplay of its neural networks means that a human can hardly check its working, or verify its decisions before they are followed through." The commentary concludes with the somewhat bleak vision that, should similar software be applied in other domains, "[t]he machine becomes an oracle; its pronouncements have to be believed." (Nature 529, 2016, p. 437)

This shows that while the techniques employed by Silver et al. (2016) lead to a fundamental breakthrough in *choosing the right moves*, they are at a loss when it comes to *explaining* and *justifying* the chosen moves. Informally speaking, the AlphaGo software is really good at playing Go, but it "cannot tell us" why (and perhaps "does not know" why). On the other hand, if

a human Go player was asked why they chose a particular move, they could most probably think of a reason and verbalise it in a way that is accessible (or can be made accessible through further explanation) to their fellow humans. Moreover, if natural and artificial intelligences (i.e., humans and computers) are ever to cooperate on complex problem-solving tasks on a basis of mutual trust, then it would bear enormous potential for beneficial collaboration if the machines could explain and justify their actions in a rational discourse.

The formal study of rational discourse is the topic of *argumentation theory*. Argumentation is concerned with the exchange of and interplay between "arguments", where it seems that the notion of what an "argument" is still lacks unanimity. We give the definition of Walton (2009):

> "There are differences in the literature in argumentation theory on how to define an argument. Some definitions are more minimal while others are more inclusive. We start here with a minimal definition, however, that will fit the introduction to the elements of argumentation presented below. *An argument is a set of statements (propositions), made up of three parts, a conclusion, a set of premises, and an inference from the premises to the conclusion.* An argument can be supported by other arguments, or it can be attacked by other arguments, and by raising critical questions about it."

So according to this definition (emphasis mine), an argument has an internal structure that links *premises* and a *conclusion* via an *inference*. Equally importantly, to Walton, an argument virtually never stands alone. It is only the interaction with other arguments (and so-called "critical questions") that ultimately determines the evaluation of a particular proposition at issue. More concretely, Walton (2009) distinguishes between argumentation and logic thus:

> "The general approach or methodology of argumentation can be described as distinctively different from the traditional approach based on deductive logic. The traditional approach concentrated on a single inference, where the premises and conclusion are designated in advance, and applied formal models like propositional calculus and quantification theory determine whether the conclusion conclusively follows from the premises. This approach is often called monological.
>
> "In contrast, the argumentation approach is called dialogical (or dialectical) in that it looks at two sides of an argument, the pro and the contra. According to this approach, the method of evaluation is to examine how the strongest arguments for and against a particular proposition at issue interact with each other, and in particular how each argument is subject to probing critical questioning that reveals doubts about it. By this dialog process of pitting the one argument against the other, the weaknesses in each argument are revealed, and it is shown which of the two arguments is the stronger."

So while the job of logic is the inference within the argument, the job of argumentation also encompasses the interaction between different arguments.

Some approaches in argumentation theory even go as far as removing the internals of arguments completely from the agenda: In the last decade, abstract argumentation frameworks (AFs; Dung, 1995) have become increasingly popular in the argumentation community (Bench-Capon and Dunne, 2007). An AF can be seen as a directed graph where the nodes are "arguments" whose internal structure is not further analysed, and where an edge from a to b expresses that argument a "attacks" argument b. While the conceptual simplicity of AFs (requiring only first-year computer science undergraduate knowledge to understand them) might account for a large part of their popularity, an often employed justification for justifying research into them is that more concrete argumentation languages can be translated into AFs and

thereby evaluated. Several languages have indeed been translated to AFs: the Carneades (Gordon, Prakken, and Walton, 2007) formalism for structured argumentation (Van Gijzel and Prakken, 2011); Caminada and Amgoud (2007) and Wyner, Bench-Capon, and Dunne (2013) translate rule-based defeasible theories into AFs.

Notwithstanding the literary success of AFs, their expressive capabilities are somewhat limited, as has been recognised many times in the literature: often it is inadequate to model argumentation scenarios having as only means of expression arguments *attacking* each other. There have been several proposals towards generalising AFs (Brewka, Polberg, and Woltran, 2014). To cite only a few examples: Prakken and Sartor (1999) add *priorities* amongst arguments that are constructed from prioritised logic programming rules; Nielsen and Parsons (2006) introduced attacks from *sets* of arguments; Cayrol and Lagasquie-Schiex (2009) presented bipolar argumentation frameworks, in which arguments can also *support* each other; and Modgil (2009) proposed attacks on *attacks* with the aim of reasoning about preferences on the object level.

As a general way to overcome the restrictions of Dung's AFs, Brewka and Woltran (2010) introduced abstract dialectical frameworks (ADFs). ADFs are – in a sense – even more abstract than AFs: while in AFs arguments are abstract and the relation between arguments is fixed to attack, in ADFs also the relations are abstract (and called *links*). The relationship between different arguments (called *statements* in ADFs) is specified by *acceptance conditions*. These are Boolean functions indicating the conditions under which a statement s can be accepted when given the acceptance status of all statements with a direct link to s (its *parents*). These parents are the statements which have a say on whether the statement in question can or must (not) be accepted. In this way, AFs are recovered in the language of ADFs by specifying for each statement the acceptance condition "accept if and only if none of the attackers is accepted."

This thesis is largely concerned with ADFs. We study their semantics, their computational complexity, their representational capabilities and finally show how they can be used (like AFs) as target languages for translations from more concrete formalisms. We also place ADFs in the bigger picture of knowledge representation formalisms by comparing them to "close relatives" such as abstract argumentation frameworks, logic programs, and propositional logic. As part of the ADF success story, we just mention a reconstruction of the Carneades model of argument (Brewka and Gordon, 2010), an instantiation of simple defeasible theories into ADFs (see Chapter 6), and recent applications of ADFs for legal reasoning and reasoning with cases by Al-Abdulkarim, Atkinson, and Bench-Capon (2014, 2015).

The present document collects and unifies several refereed papers that recently appeared in international journals (see also the next section). Owing to this, each of the chapters contains its own motivational section, and some chapters also contain concluding remarks. The main theme of this thesis is – put succinctly – the development and analysis of abstract dialectical frameworks as a formalism for representing knowledge about the interrelationship between arguments.

1.1 Publications

The results of this thesis have appeared in (or as parts of) the following publications:

- Hannes Strass. Approximating operators and semantics for abstract dialectical frameworks. *Artificial Intelligence*, 205:39–70, December 2013. (All results appear in Chapter 3.)
- Gerhard Brewka, Stefan Ellmauthaler, Hannes Strass, Johannes Peter Wallner, and Stefan Woltran. Abstract dialectical frameworks revisited. In *Proceedings of the Twenty-Third*

International Joint Conference on Artificial Intelligence (IJCAI), pages 803–809. IJCAI/AAAI, August 2013. (Some results appear in Chapter 3.)

- Hannes Strass and Johannes Peter Wallner. Analyzing the computational complexity of abstract dialectical frameworks via approximation fixpoint theory. *Artificial Intelligence*, 226:34–74, 2015. (Most results appear in Chapter 4, some results in Chapter 3. The complexity results for ultimate admissible, preferred and grounded semantics in that paper have been obtained by Johannes Wallner and are not part of this thesis. Several of the remaining results have been independently obtained by Johannes Wallner and myself, namely the ultimate-semantics parts of Lemma 4.26, Corollaries 4.27 and 4.32, and Proposition 4.30; see also the overview in Table 4.1 in Section 4.2.4.)
- Hannes Strass. Instantiating rule-based defeasible theories in abstract dialectical frameworks and beyond. *Journal of Logic and Computation*, February 2015c. Advance Access published 11 February 2015, `http://dx.doi.org/10.1093/logcom/exv004`. (All results appear in Chapter 6.)
- Hannes Strass. Expressiveness of two-valued semantics for abstract dialectical frameworks. *Journal of Artificial Intelligence Research*, 54:193–231, 2015a. (All results appear in Chapter 5.)
- Hannes Strass. The relative expressiveness of abstract argumentation and logic programming. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI)*, pages 1625–1631, Austin, TX, USA, January 2015b. (The earlier conference version of and therefore fully subsumed by the previous item.)

The research reported on herein also contributed to or influenced the following works:

- Stefan Ellmauthaler and Hannes Strass. The DIAMOND System for Computing with Abstract Dialectical Frameworks. In Simon Parsons, Nir Oren, and Chris Reed, editors, *Proceedings of the Fifth International Conference on Computational Models of Argument (COMMA)*, volume 266 of *FAIA*, pages 233–240, The Scottish Highlands, Scotland, United Kingdom, September 2014. IOS Press. (Provides an implementation of ADF reasoning and makes use of complexity results provided herein.)
- Sarah A. Gaggl and Hannes Strass. Decomposing Abstract Dialectical Frameworks. In Simon Parsons, Nir Oren, and Chris Reed, editors, *Proceedings of the Fifth International Conference on Computational Models of Argument (COMMA)*, volume 266 of *FAIA*, pages 281–292. IOS Press, September 2014. (Defines additional semantics for ADFs using the ones defined herein.)
- Sarah Alice Gaggl, Sebastian Rudolph, and Hannes Strass. On the computational complexity of naive-based semantics for abstract dialectical frameworks. In Qiang Yang and Michael Wooldridge, editors, *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2985–2991, Buenos Aires, Argentina, July 2015. IJCAI/AAAI. (Analyses the computational complexity of the asymmetric family of ultimate conflict-free semantics.)
- Marc Denecker, Gerhard Brewka, and Hannes Strass. A formal theory of justifications. In Francesco Calimeri, Giovambattista Ianni, and Mirosław Truszczyński, editors, *Proceedings of the Thirteenth International Conference on Logic Programming and Non-monotonic Reasoning (LPNMR)*, pages 250–264, Lexington, KY, USA, September 2015. Springer-Verlag Berlin Heidelberg. (Uses the embedding of AFs into approximation fixpoint theory.)

- Mario Alviano, Wolfgang Faber, and Hannes Strass. Boolean functions with ordered domains in answer set programming. In Dale Schuurmans and Michael Wellman, editors, *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI)*, pages 879–885, Phoenix, AZ, USA, February 2016. (Applies bipolar Boolean functions to logic programs with aggregates.)
- Ringo Baumann and Hannes Strass. On the number of bipolar Boolean functions. *Journal of Logic and Computation*, 2017. doi: 10.1093/logcom/exx025. Advance Access Online 07 August 2017. (Analyses the number of bipolar Boolean functions in n arguments.)

Chapter 2

Background

This chapter introduces the notation and the fundamental technical concepts we will use throughout the thesis. All defined notions of this thesis are typeset in *italics*, and an index-like list of page references to the definitions can be found at the end of the document. (There, we also list how continued examples relate to each other.)

2.1 Mathematical Notation

- $\emptyset$ is the empty set;
- $\{x \mid P(x)\}$ is the set of all x for which $P(x)$ holds;
- $\in, \cup, \cap$ denote set membership, union and intersection; $\dot\cup$ denotes disjoint union;
- $\subseteq, \supseteq$ denote sub- and superset relations (allowing set equality);
- $\subsetneq, \supsetneq$ denote *proper* sub- and superset relations (disallowing set equality)
- $|S|$ denotes the cardinality of the set S;
- $A \times B$ is the Cartesian product of sets A and B – the set of pairs $\{(a, b) \mid a \in A, b \in B\}$;
- for a set S, the powerset of S (the set of all of its subsets) is denoted by 2^S;
- $f : D_1 \times \cdots \times D_n \to R$ denotes an n-ary function f with domains $D_1, \ldots, D_n$ and range R;
- $\mathbb{N}$ denotes the set of natural numbers including zero.

2.2 Logic in Knowledge Representation and Reasoning

Propositional logic For a set A of atomic propositions (atoms, propositional variables), the set of propositional formulas φ over A is defined inductively:

$$\varphi ::= \top \mid \bot \mid a \in A \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi$$

Additional syntactical connectives can be introduced as abbreviations:

$$\begin{aligned}
\phi \to \psi &= \neg\phi \vee \psi && \text{(implication)}\\
\phi \leftrightarrow \psi &= (\phi \to \psi) \wedge (\psi \to \phi) && \text{(equivalence)}\\
\phi \mathbin{\dot\vee} \psi &= \neg(\phi \leftrightarrow \psi) && \text{(exclusive disjunction)}\\
\phi \downarrow \psi &= \neg(\phi \vee \psi) && \text{(joint denial)}
\end{aligned}$$

The semantics of propositional logic is model-theoretic. A *two-valued interpretation* is a function $v : A \to \{\mathbf{t}, \mathbf{f}\}$. The homomorphic extension $\tilde{v}$ on formulas over A is defined by induction on the structure of formulas:

$$\begin{aligned}
\tilde{v}(\top) &= \mathbf{t}\\
\tilde{v}(\bot) &= \mathbf{f}\\
\tilde{v}(a) &= v(a) && \text{for } a \in A\\
\tilde{v}(\neg\phi) &= \begin{cases} \mathbf{t} & \text{if } \tilde{v}(\phi) = \mathbf{f} \\ \mathbf{f} & \text{otherwise} \end{cases}\\
\tilde{v}(\phi \wedge \psi) &= \begin{cases} \mathbf{t} & \text{if } \mathbf{f} \notin \{\tilde{v}(\phi), \tilde{v}(\psi)\} \\ \mathbf{f} & \text{otherwise} \end{cases}\\
\tilde{v}(\phi \vee \psi) &= \begin{cases} \mathbf{t} & \text{if } \mathbf{t} \in \{\tilde{v}(\phi), \tilde{v}(\psi)\} \\ \mathbf{f} & \text{otherwise} \end{cases}
\end{aligned}$$

In a slight abuse of notation, we usually write $v(\phi)$ instead of $\tilde{v}(\phi)$.

We conveniently represent two-valued interpretations by sets $M \subseteq A$ with the understanding that M defines the interpretation

$$v_M : A \to \{\mathbf{t}, \mathbf{f}\} \qquad \text{with} \qquad a \mapsto \begin{cases} \mathbf{t} & \text{if } a \in M \\ \mathbf{f} & \text{otherwise} \end{cases}$$

We then write $M \models \varphi$ if and only if $v_M(\varphi) = \mathbf{t}$.

We sometimes modify propositional formulas by replacing some of their atoms by truth constants. For a formula φ over a vocabulary A and an interpretation $v : A \to \{\mathbf{t}, \mathbf{f}\}$, we denote by $\varphi[a/\bot : v(a) = \mathbf{f}]$ the formula obtained from φ by replacing all occurrences of atoms $a \in A$ that are false in v by the formula $\bot$. We use a similar notation for $\top$ and combinations of the two.

Logic programming For propositional logic programming, we use a nonempty set A, the *signature*, or set of *atoms*. For such an A, we define *not* $A = \{\mathit{not}\ a \mid a \in A\}$, and the set of *negation-as-failure literals* over A as $A^{\pm} = A \cup \mathit{not}\ A$. Likewise, we define *not not* $A = \{\mathit{not\ not}\ a \mid a \in A\}$ as the set of *doubly-negated logic program literals* and consequently $A^{\pm-} = A^{\pm} \cup \mathit{not\ not}\ A$.

A *canonical logic program rule* over A is then of the form $a \leftarrow M$ where $a \in A$ and $M \subseteq A^{\pm-}$. A *normal logic program rule* over A is a canonical logic program rule where $M \subseteq A^{\pm}$. A normal logic program rule can be read as logical consequence, "a is true if all literals in M are true." We denote by $M^+ = M \cap A$ and $M^- = \{a \in A \mid \mathit{not}\ a \in M\}$ the *positive* and *negative body* atoms, respectively. A rule is *definite* iff $M \subseteq A$. For singleton $M = \{m\}$ we denote the rule just by $a \leftarrow m$. A *logic program (LP)* P over A is a set of logic program rules over A, and it is normal (definite) if all rules in it are normal (definite). By default, we will mean normal logic programs when just saying "logic program", and will use the addendum "canonical" explicitly.

Nonmonotonic reasoning There are further knowledge representation formalisms that are relevant for this thesis, but will not feature as prominently as the ones defined above, for example the default logic of Reiter (1980) and the autoepistemic logic of Moore (1985). We will introduce them in more detail later in this work.

2.3 Lattice Theory

Let us now recall some basic concepts from lattice theory.

Partially ordered set A *partially ordered set (poset)* is a pair $(L, \sqsubseteq)$ where $\sqsubseteq$ is a binary relation on L that is reflexive, transitive, and antisymmetric.

Monotone operator An operator $O : L \to L$ is *monotone* if for all $x \sqsubseteq y$ we find $O(x) \sqsubseteq O(y)$; it is *antimonotone* if for all $x \sqsubseteq y$ we find $O(y) \sqsubseteq O(x)$.

Fixpoint An $x \in L$ is a *fixpoint* of O iff $O(x) = x$; an $x \in L$ is a *prefixpoint* of O iff $O(x) \sqsubseteq x$ and a *postfixpoint* of O iff $x \sqsubseteq O(x)$.

Complete lattice A *complete lattice* is a partially ordered set (poset) $(L, \sqsubseteq)$ where every subset S of L has a least upper bound $\bigsqcup S \in L$ and a greatest lower bound $\bigsqcap S \in L$. In particular, a complete lattice has a least ($\bot$) and a greatest ($\top$) element. When dealing with different structures at the same time, we sometimes index $\bigsqcup, \bigsqcap, \bot, \top$ to indicate to which structure they belong. For example, $\bot_L$ refers to the $\sqsubseteq$-least element of the lattice $(L, \sqsubseteq)$.[1]

Due to a fundamental result by Tarski and Knaster (Tarski, 1955), for any monotone operator O on a complete lattice, the set of its fixpoints forms a complete lattice itself (Davey and Priestley, 2002, Theorem 2.35). In particular, its least fixpoint $lfp(O)$ exists; additionally, the least prefixpoint of O is also its least fixpoint.

In this thesis, we will also be concerned with further, more general algebraic structures.

Complete meet-semilattice A complete meet-semilattice is a partially ordered set $(L, \sqsubseteq)$ where every non-empty subset $\emptyset \neq S \subseteq L$ has a greatest lower bound (meet) $\bigsqcap S \in L$ and every directed subset $D \subseteq C$ has a least upper bound $\bigsqcup D \in C$. A set is directed iff it is nonempty and each pair of elements has an upper bound in the set. Every complete lattice is a complete meet-semilattice, but not vice versa. (A complete meet semi-lattice need not have a greatest element.)

Complete partially ordered sets (CPOs) A CPO is a partially ordered set $(C, \leq)$ with a $\leq$-least element where each directed subset $D \subseteq C$ has a least upper bound $\bigsqcup D \in C$. Clearly every complete meet-semilattice is a complete partially ordered set, but not necessarily vice versa. (A complete partial order need not have greatest lower bounds.) Fortunately, complete partially ordered sets still guarantee the existence of (least) fixpoints for monotone operators.

Theorem 2.1 (Davey and Priestley, 2002, Theorem 8.22**).** *In a complete partially ordered set $(C, \leq)$, any $\leq$-monotone operator $O : C \to C$ has a least fixpoint.*

[1] Although some notational confusion of least and greatest lattice elements, respectively, and the syntactic truth and falsity symbols from propositional logic could arise, we hope that it will be clear from the context which is meant.

2.4 Approximation Fixpoint Theory

Building upon Knaster and Tarski's fundamental result, Denecker, Marek, and Truszczyński (2000) introduce the important concept of an approximation of an operator. In the study of semantics of knowledge representation formalisms, elements of lattices represent objects of interest. Operators on lattices transform such objects into others according to the contents of some knowledge base. Consequently, fixpoints of such operators are then objects that cannot be updated any more – informally speaking, the knowledge base can neither add information to a fixpoint nor remove information from it.

To study fixpoints of operators O, DMT study fixpoints of their *approximating operators* $\mathcal{O}$.[2] When O operates on a set L, its approximation $\mathcal{O}$ operates on pairs $(x, y) \in L^2$ where L^2 denotes $L \times L$. Such a pair can be seen as representing a *set* of lattice elements by providing a lower bound x and an upper bound y. Consequently, the pair (x, y) approximates all $z \in L$ such that $x \sqsubseteq z \sqsubseteq y$. Of special interest are *consistent* pairs – those where $x \sqsubseteq y$, that is, the set of approximated elements is nonempty. A pair (x, y) with $x = y$ is called *exact* – it "approximates" a single element of the original lattice.[3]

There are two natural orderings on approximating pairs: first, the *information ordering* $\leq_i$, that intuitively orders pairs according to their information content.[4] Formally, for four elements $x_1, x_2, y_1, y_2 \in L$ we define

$$(x_1, y_1) \leq_i (x_2, y_2) \text{ iff } x_1 \sqsubseteq x_2 \text{ and } y_2 \sqsubseteq y_1.$$

This ordering leads to a complete lattice $(L^2, \leq_i)$, the product of L with itself, its *bilattice*. For example, the pair $(\bot, \top)$ consisting of $\sqsubseteq$-least $\bot$ and $\sqsubseteq$-greatest lattice element $\top$ approximates all lattice elements and thus contains no information – it is the least element of the bilattice $(L^2, \leq_i)$; exact pairs (x, x) are those that are maximally informative while still being consistent. The second natural ordering is the *truth ordering* $\leq_t$, which orders elements of the bilattice according to their degree of truth. Formally, for $x_1, x_2, y_1, y_2 \in L$ it is defined by

$$(x_1, y_1) \leq_t (x_2, y_2) \text{ iff } x_1 \sqsubseteq x_2 \text{ and } y_1 \sqsubseteq y_2.$$

The pair $(\bot, \bot)$ is the least element of $\leq_t$ – in a truth-based setting, it assigns the truth value false to all elements of L; the pair $(\top, \top)$ consequently is the $\leq_t$-greatest element – here, all elements of L are assigned value true.

To define an approximation operator $\mathcal{O} : L^2 \to L^2$, one essentially has to define two functions: a function $\mathcal{O}' : L^2 \to L$ that yields a new *lower* bound (first component) for a given pair; and a function $\mathcal{O}'' : L^2 \to L$ that yields a new *upper* bound (second component) for a given pair. Accordingly, the overall approximation is then given by $\mathcal{O}(x, y) = (\mathcal{O}'(x, y), \mathcal{O}''(x, y))$ for $(x, y) \in L^2$. Conversely, in case $\mathcal{O}$ is considered given, the notations $\mathcal{O}'(x, y)$ and $\mathcal{O}''(x, y)$ are read as the projection of $\mathcal{O}(x, y)$ to the first and second component, respectively.

Denecker et al. (2000) identify an important subclass of operators on bilattices, namely those that are *symmetric*, that is, for which $\mathcal{O}'(x, y) = \mathcal{O}''(y, x)$. For these, $\mathcal{O}(x, y) = (\mathcal{O}'(x, y), \mathcal{O}'(y, x))$, and to define $\mathcal{O}$ it suffices to specify $\mathcal{O}'$. An operator now is *approximating* if it is symmetric and $\leq_i$-monotone. For an antimonotone operator O, its *canonical approximating operator* $\mathcal{O}$ is given by $\mathcal{O}'(x, y) = (O(y), O(x))$.

The main contribution of Denecker et al. (2000) was the association of the *stable operator* $\mathcal{SO}$ to an approximating operator $\mathcal{O}$. Below, the expression $\mathcal{O}'(\cdot, y) : L \to L$ denotes the operator given by $x \mapsto \mathcal{O}'(x, y)$ for $x \in L$.

[2] The approximation of an operator O is typographically indicated by a calligraphic $\mathcal{O}$.

[3] Denecker et al. (2000) call such pairs "complete," we however use that term for argumentation in a different meaning and want to avoid confusion.

[4] The ordering is also called *knowledge* ordering or *precision* ordering (Denecker et al., 2000; Fitting, 2002).

Definition 2.1. For a complete lattice $(L, \sqsubseteq)$ and an approximating operator $\mathcal{O} : L^2 \to L^2$, define the

- *complete stable operator for* $\mathcal{O}$ as $c\mathcal{O} : L \to L$ by $c\mathcal{O}(y) = lfp(\mathcal{O}'(\cdot, y))$;
- *stable operator for* $\mathcal{O}$ as $\mathcal{SO} : L^2 \to L^2$ by $\mathcal{SO}(x, y) = (c\mathcal{O}(y), c\mathcal{O}(x))$. ◇

This general, lattice-theoretic definition by DMT yields a uniform treatment of the standard semantics of the major nonmonotonic knowledge representation formalisms – logic programming, default logic and autoepistemic logic (Denecker, Marek, and Truszczyński, 2003).

Definition 2.2. Let $(L, \sqsubseteq)$ be a complete lattice and $\mathcal{O} : L^2 \to L^2$ be an approximating operator. Furthermore, let $x, y \in L$ with $x \sqsubseteq y$. Define the following semantical notions for $\mathcal{O}$:

Kripke-Kleene semantics	$lfp(\mathcal{O})$
three-valued supported model (x, y)	$\mathcal{O}(x, y) = (x, y)$
two-valued supported model (x, x)	$\mathcal{O}(x, x) = (x, x)$
well-founded semantics	$lfp(\mathcal{SO})$
three-valued stable model (x, y)	$\mathcal{SO}(x, y) = (x, y)$
two-valued stable model (x, x)	$\mathcal{SO}(x, x) = (x, x)$

◇

It is clear that each two-valued supported/stable model is a three-valued supported/stable model; furthermore the Kripke-Kleene semantics of an operator is a three-valued supported model and the well-founded semantics is a three-valued stable model. Also, each three-valued/two-valued stable model is a three-valued/two-valued supported model, which is easily seen: if (x, y) is a three-valued stable model, we have $(x, y) = \mathcal{SO}(x, y)$. Now $(x, y) = \mathcal{SO}(x, y) = (c\mathcal{O}(y), c\mathcal{O}(x)) = (lfp(\mathcal{O}'(\cdot, y)), lfp(\mathcal{O}'(\cdot, x)))$ implies $x = \mathcal{O}'(x, y)$ and $y = \mathcal{O}'(y, x)$, whence $(x, y) = (\mathcal{O}'(x, y), \mathcal{O}'(y, x)) = \mathcal{O}(x, y)$ and (x, y) is a three-valued supported model. This holds in particular if $x = y$, and each two-valued stable model is a two-valued supported model.

Ultimate approximations In subsequent work, Denecker, Marek, and Truszczyński (2004) presented a general, abstract way to define the most precise approximation of a given operator O in a lattice $(L, \sqsubseteq)$. Most precise here refers to a generalisation of $\leq_i$ to operators, where for $\mathcal{O}_1, \mathcal{O}_2 : L^2 \to L^2$, they define

$$\mathcal{O}_1 \leq_i \mathcal{O}_2 \text{ iff for all } x, y \in L \text{ with } x \sqsubseteq y \text{ it holds that } \mathcal{O}_1(x, y) \leq_i \mathcal{O}_2(x, y).$$

For consistent pairs (x, y) of the bilattice $(L^2, \leq_i)$, they show that the most precise – called the *ultimate* – approximation of O is given by $\mathcal{U}_O(x, y) = \big(\mathcal{U}'_O(x, y), \mathcal{U}''_O(x, y)\big)$ with

$$\mathcal{U}'_O(x, y) = \bigsqcap \{O(z) \mid x \sqsubseteq z \sqsubseteq y\}$$
$$\mathcal{U}''_O(x, y) = \bigsqcup \{O(z) \mid x \sqsubseteq z \sqsubseteq y\}$$

Note that the ultimate approximation works only for consistent pairs and is not symmetric. However, this is not of harm for two-valued stable models: a pair (x, x) is an ultimate two-valued stable model of O iff $x = lfp(\mathcal{U}'_O(\cdot, x))$; here it suffices that the ultimate approximation is defined for consistent pairs only, as we have

$$\bot \sqsubseteq \mathcal{U}'_O(\bot, x) \sqsubseteq \mathcal{U}'_O(\mathcal{U}'_O(\bot, x)) \sqsubseteq \ldots \sqsubseteq \mathcal{U}'_O(x, x) \sqsubseteq x.$$

The definition of the ultimate approximation operator is quite remarkable since previously, approximating operators $\mathcal{O}$ for lattice operators O had to be devised by hand rather than automatically derived. We next illustrate the workings of the operator-based framework for the case of logic programming.

2.4.1 Logic Programming

To show how approximation fixpoint theory applies to normal logic programs, we use definitions along the lines of Fitting (2002), whose fixpoint-theoretic approach to logic programming was extended by Denecker et al. (2000). The perhaps most prominent example for an operator is the one-step consequence operator T_P associated with a definite logic program P (Fitting, 2002). For a signature A, it operates on subsets of A and assigns to a set of atoms S those atoms which are implied by S according to the rules in P. The underlying lattice is therefore $(2^A, \subseteq)$ consisting of the set of A's subsets ordered by $\subseteq$.

This operator was later generalised to four-valued Belnap logic (Fitting, 2002) and can be recast in a bilattice-based setting as follows. A pair $(X, Y) \in 2^A \times 2^A$ can be read as a four-valued assignment by evaluating all atoms in $X \cap Y$ as true, those in $A \setminus (X \cup Y)$ as false, the ones in $Y \setminus X$ as undefined and the atoms in $X \setminus Y$ as inconsistent.

Definition 2.3. For a normal logic program P over A, define an (approximating) operator $\mathcal{T}_P : 2^A \times 2^A \to 2^A \times 2^A$ as follows: for $X, Y \subseteq A$,

$$\mathcal{T}_P(X, Y) = (\mathcal{T}'_P(X, Y), \mathcal{T}'_P(Y, X))$$
$$\mathcal{T}'_P(X, Y) = \left\{ a \in A \;\middle|\; a \leftarrow M \in P, M^+ \subseteq X, M^- \cap Y = \emptyset \right\} \qquad \diamond$$

Roughly, to construct a new lower bound, the operator $\mathcal{T}'_P$ returns all those atoms for which a rule exists whose positive body is implied by the current lower bound and whose negative body does not share an atom with the current upper bound. This first of all means that the operator allows to infer an atom via a program rule if – according to the input estimate – the positive body is true and the negative body is false. The fixpoints of $\mathcal{T}_P$ are the four-valued *supported models* of P; its consistent fixpoints are the three-valued supported models of P. The two-valued supported models of P are computed by the abovementioned operator T_P, that – in this setting – is defined by $T_P(M) = \mathcal{T}'_P(M, M)$ (Denecker et al., 2000).

The abstract principles of Denecker et al. (2000) outlined above also yield the corresponding *stable* operator $\mathcal{ST}_P$. This operator in turn immediately yields the Gelfond-Lifschitz operator $GL_P(M) = \mathcal{ST}'_P(M, M)$ for computing two-valued stable models of P. The stable operator $\mathcal{ST}_P$ also gives rise to the *well-founded model* of P, which is the least fixpoint of $\mathcal{ST}_P$. Additionally, *three-valued stable models* are the consistent fixpoints of $\mathcal{ST}_P$. These are further refined into two additional semantics: *M-stable models* are three-valued stable models (X, Y) where X is $\subseteq$-maximal – M-stable is for "maximal stable" (Saccà and Zaniolo, 1997); *L-stable models* are three-valued stable models (X, Y) where $Y \setminus X$ is $\subseteq$-minimal – L-stable is for "least undefined" (Saccà and Zaniolo, 1997). It is clear that these same maximisation/minimisation criteria can be applied to consistent fixpoints of $\mathcal{T}_P$ – the three-valued supported models. This leads to *M-supported models* and *L-supported models*. In a table much like the one from Definition 2.2, this looks thus:

M-supported model (X, Y)	$\mathcal{T}_P(X, Y) = (X, Y)$ and (X, Y) is $\leq_i$-maximal
L-supported model (X, Y)	$\mathcal{T}_P(X, Y) = (X, Y)$ and $Y \setminus X$ is $\subseteq$-minimal
M-stable model (X, Y)	$\mathcal{ST}_P(X, Y) = (X, Y)$ and (X, Y) is $\leq_i$-maximal
L-stable model (X, Y)	$\mathcal{ST}_P(X, Y) = (X, Y)$ and $Y \setminus X$ is $\subseteq$-minimal

It follows that each two-valued supported/stable model is an L-supported/L-stable model is an M-supported/M-stable model is a three-valued supported/stable model.

As an example, consider the logic program $P_1 = \{a \leftarrow \emptyset, b \leftarrow a\}$. It is a definite LP, thus we can iterate its two-valued one-step consequence operator T_{P_1} on the empty set, the least element of the relevant lattice: we have $T_{P_1}(\emptyset) = \{a\}$ and $T_{P_1}(\{a\}) = \{a, b\} = T_{P_1}(\{a, b\})$ as a

fixpoint and thus the least (two-valued supported) model of program P_1. Now we add another rule to this program and set $P_2 = P_1 \cup \{c \leftarrow \{b, not\ d\}\}$, a logic program over $A = \{a, b, c, d\}$ that is not definite. To compute its well-founded model, we iterate the associated stable four-valued one-step consequence operator $\mathcal{ST}_{P_2}$ on the least element $(\emptyset, A)$ of the relevant bilattice. We see that $\mathcal{ST}_{P_2}(\emptyset, A) = (\{a\}, \{a, b, c\})$: intuitively, a is added to the lower bound since its body is satisfied, d is removed from the upper bound because there is no program rule to derive d. Applying $\mathcal{ST}_{P_2}$ again leads to the pair $(\{a, b, c\}, \{a, b, c\})$ which is an exact fixpoint and thus the only two-valued stable model of P_2.

2.5 Abstract Argumentation

2.5.1 Abstract Argumentation Frameworks

Dung (1995) introduced a way to study the "fundamental mechanisms that humans use in argumentation". His argumentation frameworks (AFs) F are pairs (A, R) where A is a set and $R \subseteq A \times A$; to access the components of an AF $F = (B, S)$, we use the notations $A_F = B$ and $R_F = S$. The intended reading of an AF F is that the elements of A are arguments whose internal structure is abstracted away. The only information about the arguments is given by the relation R encoding a notion of attack: for $a, b \in A$ a pair $(a, b) \in R$ expresses that argument a attacks argument b in some sense. This seemingly lightweight formalism allows for a rich semantical theory, whose most important notions we subsequently recall.

The purpose of semantics for argumentation frameworks is to determine sets of arguments which are acceptable according to various standards. As an intuitive example, a set of arguments could be accepted if it is internally consistent and can defend itself against attacks from the outside. More formally, a set $S \subseteq A$ of arguments is *conflict-free* iff there are no $a, b \in S$ with $(a, b) \in R$. For an argument $a \in A$, the set of its attackers is $R_F^{-1}(a) = \{b \in A \mid (b, a) \in R_F\}$. An AF is *finitary* iff $R_F^{-1}(a)$ is finite for all $a \in A$. For $S \subseteq A$, the set of arguments it attacks is $R_F(S) = \bigcup_{a \in S} R_F(a) = \{b \in A \mid (a, b) \in R_F \text{ for some } a \in S\}$. Finally, for $S \subseteq A$ and $a \in A$, the set S *defends* a iff $R_F^{-1}(a) \subseteq R_F(S)$, that is, all attackers of a are attacked by S.

The major semantics for argumentation frameworks can be formulated using two operators that Dung (1995) already studied. The first is the *characteristic function of an AF* F: for $S \subseteq A$, define $D_F(S) = \{a \in A \mid S \text{ defends } a\}$. This operator D_F is $\subseteq$-monotone and therefore has a least fixpoint in the lattice $(2^A, \subseteq)$. This least fixpoint of D_F is defined as the *grounded* extension of F. The second relevant operator U_F takes as input a set S of arguments, and returns the arguments that are not attacked by any argument in S (U is for "unattacked") – formally $U_F(S) = A \setminus R_F(S)$. It is an antimonotone operator, and its fixpoints are the *stable* extensions of F. Additionally, U_F can characterise conflict-freeness: a set $S \subseteq A$ is conflict-free iff $S \subseteq U_F(S)$. Further semantics are defined as follows. A set $E \subseteq A$ is a *complete* extension iff it is a conflict-free fixpoint of D_F. More generally, a set $S \subseteq A$ is *admissible* iff S is conflict-free and $S \subseteq D_F(S)$. Finally, *preferred* extensions are $\subseteq$-maximal complete extensions; and *semi-stable* extensions are those complete extensions E where the set $E \cup R_F(E)$ (the *range* of the extension E) is $\subseteq$-maximal. The same maximisation criteria that lead from admissible sets to preferred and semi-stable extensions can also be applied to conflict-free sets: a *naive extension of an AF* is a $\subseteq$-maximal conflict-free set; a *stage extension of an AF* is a conflict-free set with $\subseteq$-maximal range. For two argumentation frameworks $F_1 = (A_1, R_1)$ and $F_2 = (A_2, R_2)$, their union is defined as $F_1 \cup F_2 = (A_1 \cup A_2, R_1 \cup R_2)$.

Example 2.1. Let the argumentation framework $F = (A, R)$ be given by $A = \{a, b, c, d\}$ and $R = \{(a, b), (c, d), (d, c)\}$. It is depicted by the following directed graph:

Its grounded extension is the set $G = \{a\}$; it possesses two stable extensions, $E_1 = \{a, c\}$ and $E_2 = \{a, d\}$. The three sets G, E_1, E_2 form the only complete extensions of F. ◇

2.5.2 Abstract Dialectical Frameworks

Brewka and Woltran (2010) introduced abstract dialectical frameworks as a powerful generalisation of abstract argumentation frameworks that are able to capture not only attack and support, but also more general notions such as joint attack and joint support.

Definition 2.4. An *abstract dialectical framework (ADF)* is a triple $D = (S, L, C)$ where

- S is a set of *statements*,
- $L \subseteq S \times S$ is a set of *links*, where $par(s) = \{r \in S \mid (r, s) \in L\}$
- $C = \{C_s\}_{s \in S}$ is a set of total functions $C_s : 2^{par(s)} \to \{\mathbf{t}, \mathbf{f}\}$. ◇

Intuitively, the function C_s for a statement s determines the acceptance status of s, which naturally depends on the status of its parent nodes $par(s)$. Alternatively, any such function C_s can be represented by the set of all parent subsets leading to acceptance, $C_s^{\mathbf{t}} = \{M \subseteq par(s) \mid C_s(M) = \mathbf{t}\}$. We will use both representations in this thesis and indicate the alternative one by writing an ADF as $(S, L, C^{\mathbf{t}})$.

Many more specific representations of acceptance conditions are possible. Brewka and Woltran (2010) even introduce two of these additional representations: For one, an acceptance condition C_a can be described via a propositional formula φ_a over the vocabulary $par(a)$, which is straightforward to use whenever each statement has only finitely many relevant parents. The understanding there is that $C_a^{\mathbf{t}}$ is given by the two-valued models of φ_a, where an interpretation is identified with the set of atoms that are evaluated to true. For another, Brewka and Woltran (2010) also demonstrated how assigning weights to links and combining these weights with proof standards can give rise to acceptance conditions.

Example 2.2. The following is a simple ADF: $D = (S, L, C^{\mathbf{t}})$ with statements $S = \{a, b, c, d\}$, links $L = \{(a, c), (b, b), (b, c), (b, d)\}$ and acceptance functions given by $C_a^{\mathbf{t}} = \{\emptyset\}$, $C_b^{\mathbf{t}} = \{\{b\}\}$, $C_c^{\mathbf{t}} = \{\{a, b\}\}$ and $C_d^{\mathbf{t}} = \{\emptyset\}$. These acceptance functions can intuitively be interpreted as follows:

- Statement a has no parents, $par(a) = \emptyset$, thus $2^{par(a)} = \{\emptyset\}$. The acceptance function specifies that $\emptyset \mapsto \mathbf{t}$, whence a is always $\mathbf{t}$.
- Statement b is its own parent. According to its acceptance function, it is $\mathbf{t}$ only if it is $\mathbf{t}$. Statement b is thus (cyclically) self-supporting.
- Statement c has parents $par(c) = \{a, b\}$. They jointly support c, as is witnessed by $C_c^{\mathbf{t}} = \{par(c)\}$. Note that joint support here indeed means that the support only becomes effective if *both* parents are $\mathbf{t}$.
- Statement d is attacked by its only parent b. ◇

Brewka and Woltran (2010) introduced several semantical notions for ADFs.

Definition 2.5. For an ADF $D = (S, L, C^{\mathbf{t}})$, a set $M \subseteq S$ is *conflict-free* iff for all $s \in M$ we have $M \cap par(s) \in C_s^{\mathbf{t}}$. A set $M \subseteq S$ is a *two-valued model for an ADF* (or briefly *model for an ADF*) D iff for each $s \in S$ we have $s \in M$ iff $M \cap par(s) \in C_s^{\mathbf{t}}$. ◇

We illustrate the semantics with our running example.

Example 2.3 (Continued from Example 2.2). A conflict in a set of statements intuitively means that there is either an attack within the set or a lack of support for some statement. The running example ADF D has the following conflict-free sets:

$$\emptyset, \{a\}, \{b\}, \{d\}, \{a, b\}, \{a, d\}, \{a, b, c\}$$

This is easy to understand: from all subsets of $S = \{a, b, c, d\}$, we have to remove those that (1) contain both b and d, since b attacks d; or (2) contain c without containing both a and b, because c depends on joint support of a and b. The remaining ones above are conflict-free.

The two models of D are $M_1 = \{a, b, c\}$ and $M_2 = \{a, d\}$. Intuitively, a is always $\mathbf{t}$ and thus contained in both models. For the self-supporting b, the model semantics has a choice whether or not to accept it, and this choice determines the two models. In M_1, statement b is accepted along with a, their joint support of c becomes relevant and c is also accepted. (Statement d is not accepted by M_1 since b is accepted and attacks d.) In M_2, statement b is not accepted whence c is not accepted due to a lack of support; statement d behaves like an AF argument and thus is accepted because its only attacker b is not accepted. ◇

Some semantics were only defined for a subclass of ADFs called *bipolar*. Intuitively, in bipolar abstract dialectical frameworks (BADFs) each link is supporting or attacking (or both); that is, there are no links that sometimes support and sometimes attack (depending on the values of other parents). The formal definition follows.

Definition 2.6. Let $D = (S, L, C)$ be an ADF.

- A link $(r, s) \in L$ is *supporting* in D iff for all $M \subseteq par(s)$, we have that $M \in C_s^{\mathbf{t}}$ implies $M \cup \{r\} \in C_s^{\mathbf{t}}$;
- symmetrically, a link $(r, s) \in L$ is *attacking* in D iff for all $M \subseteq par(s)$, we have that $M \cup \{r\} \in C_s^{\mathbf{t}}$ implies $M \in C_s^{\mathbf{t}}$.
- An ADF $D = (S, L, C)$ is *bipolar* iff all links in L are supporting or attacking; we use L^+ to denote all supporting and L^- to denote all attacking links of L in a bipolar ADF. ◇

Brewka and Woltran (2010) defined a version of the stable model semantics for bipolar ADFs: A model M of a bipolar ADF D is a *BW-stable model* of D iff it is the least model of the reduced ADF D^M defined as $D^M = (S^M, L^M, C^M)$ with

- $S^M = S \cap M$ (nodes are restricted to those in the model),
- $L^M = \{(r, s) \mid r, s \in S^M, (r, s) \in L^+\}$ (links are restricted to supporting links among nodes in the model) and
- for each $s \in S^M$ and $B \subseteq S^M$, we set $C_s^M(B) = \mathbf{t}$ iff $C_s(B) = \mathbf{t}$ (likewise the acceptance functions are restricted to the remaining parent nodes).

Stable models then serve to define further notions; but first let us define how to remove a set R of statements from an ADF $D = (S, L, C^{\mathbf{t}})$ as follows: $D - R = (\hat{S}, \hat{L}, \hat{C})$, where

- $\hat{S} = S \setminus R$ (the nodes in R are removed),
- $\hat{L} = L \cap (\hat{S} \times \hat{S})$ (links are restricted to the remaining nodes) and
- $\hat{C} = \left\{ \left\{ B \cap \hat{S} \;\middle|\; B \in C_s^{\mathbf{t}} \right\} \right\}_{s \in \hat{S}}$ (likewise, acceptance conditions are restricted to the remaining parents).

For a bipolar ADF $D = (S, L, C)$, a set $M \subseteq S$ is *BW-admissible* in D iff there is some $R \subseteq S$ with

- $L^- \cap (R \times M) = \emptyset$ (there are no attacks from R to M) and
- M is a stable model of $D - R$.

A set $M \subseteq S$ is a *BW-preferred model* of D iff it is $\subseteq$-maximal among the sets that are BW-admissible in D. Finally, Brewka and Woltran (2010) also generalise the grounded semantics: for $D = (S, L, C)$ they define a monotone operator $\Gamma_D : 2^S \times 2^S \to 2^S \times 2^S$ by $(X, Y) \mapsto (\Gamma'_D(X, Y), \Gamma''_D(X, Y))$, where[5]

$$\Gamma'_D(X, Y) = \left\{ s \in S \;\middle|\; \text{for all } X \subseteq Z \subseteq Y, \text{we have } Z \cap par(s) \in C_s^{\mathbf{t}} \right\}$$
$$\Gamma''_D(X, Y) = \left\{ s \in S \;\middle|\; \text{there exists } X \subseteq Z \subseteq Y \text{ with } Z \cap par(s) \in C_s^{\mathbf{t}} \right\}$$

The $\leq_i$-least fixpoint of Γ_D gives rise to the *BW-well-founded model* of D.

Example 2.4 (Continued from Example 2.3). The $\leq_i$-least fixpoint of Γ_D is the pair $(\{a\}, \{a, b, c, d\})$, therefore the BW-well-founded model of D is the set $\{a\}$. Intuitively, statement a is in there because it is always **t**. Statement b is not contained in the BW-well-founded model since it is only self-supporting. Statement c is not contained because it needs joint support by a and b, of which b is missing. For d, it cannot be guaranteed that its attacker b is necessarily **f**, since it is still contained in the upper bound of Γ_D's least fixpoint. ◇

It is clear that ADFs are a generalisation of AFs: for an argumentation framework $F = (A, R)$, its *associated abstract dialectical framework* is $D(F) = (A, R, C^{\mathbf{t}})$, where $C_a^{\mathbf{t}} = \{\emptyset\}$ for each $a \in A$. But this is not just syntactical; Brewka and Woltran (2010) showed that their semantical notions for ADFs are generalisations of Dung's respective AF notions:

Proposition 2.2. *Let $F = (A, R)$ be an argumentation framework and $D(F) = (A, R, C^{\mathbf{t}})$ its associated abstract dialectical framework. The following are in one-to-one correspondence:*

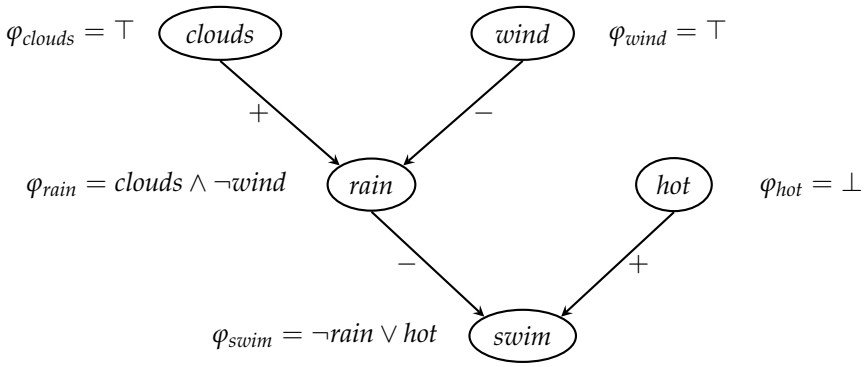
1. *the grounded extension of F and the BW-well-founded model of $D(F)$;*
2. *conflict-free sets of F and conflict-free sets of $D(F)$;*
3. *stable extensions of F and models of $D(F)$;*
4. *stable extensions of F and BW-stable models of $D(F)$;*
5. *preferred extensions of F and BW-preferred models of $D(F)$.*

Proof. Propositions 3, 1, 7 and 12 of (Brewka and Woltran, 2010). □

[5] The representation of the operator and the lattice it operates on given by Brewka and Woltran (2010) is slightly different: both representations use pairs of sets of statements to describe the current acceptance status of statements. Their pairs explicitly represent the statements that are **t** in the first component and the ones that are **f** in the second component. Since our second component explicitly represents the statements that are *not* **f**, we adjusted the definition of the operator Γ''_D for computing the second component.

It is especially notable that models and stable models coincide for AF-based ADFs, a fact that we will illuminate further and for which we will provide an intuitive explanation.

Example 2.5 (Adapted from Brewka and Woltran, 2010, Example 6). Consider a scenario where we want to decide whether we go for a *swim*. We do so if there is no *rain*, or it is *hot*. It is warm, but not hot, and there are *clouds* indicating that it might rain. However the reliable weather forecast predicts *wind* that will blow away the clouds. Using the vocabulary $S = \{clouds, wind, rain, hot, swim\}$, we devise the bipolar ADF $D_{swim} = (S, L^+ \cup L^-, C)$ shown below to model this deliberation process. Here, statements are depicted as nodes, edges represent links and acceptance conditions are written as propositional formulas next to the statements.



Supporting and attacking links are designated using the labels $+$ and $-$; this is however only for illustration as the polarity of the links can be read off the acceptance formulas. The statement *rain*, for example, is supported by the statement *clouds* and attacked by the statement *wind*. According to φ_{rain}, the attack from *wind* is stronger than the support from *clouds*. That is, as soon as we accept *wind*, we must reject *rain*. On the other hand, *swim* is attacked by rain and supported by hot. Here, by φ_{swim}, the support from *hot* is stronger than the attack from *rain*; or put another way, the missing attack from *rain* is stronger than the missing support from *hot*. This effectively means that rejecting *rain* leads to accepting *swim*. ◇

2.6 Complexity Theory

We give only a very short overview on basic concepts of complexity theory here. We refer to the textbooks by Papadimitriou (2003) and Arora and Barak (2009) for a detailed exposition.

Assume some fixed finite vocabulary Σ with $|\Sigma| > 1$. A language $L \subseteq \Sigma^*$ is in P iff it can be recognised by a deterministic Turing machine in polynomial time. Complexity class NP contains all problems $L \subseteq \Sigma^*$ that have a polytime-computable witness relation; that is, $L \in \mathsf{NP}$ iff there are $W_L \in \mathsf{P}$ and $k \in \mathbb{N}$ such that: $x \in L$ iff there is a y such that $(x, y) \in W_L$ and $|y| \leq |x|^k$. For any class $\mathcal{C}$ of languages, its complement class is $\mathsf{co}\mathcal{C} = \{\overline{L} \mid L \in \mathcal{C}\}$. For example, the class coNP contains all languages L whose complement $\overline{L} = \Sigma^* \setminus L$ is in NP. These two classes give rise to the polynomial hierarchy, that can be defined (using oracle Turing machines) as follows: $\Delta_0^P = \Sigma_0^P = \Pi_0^P = \mathsf{P}$, and for $i \geq 0$, $\Delta_{i+1}^P = \mathsf{P}^{\Sigma_i^P}$, $\Sigma_{i+1}^P = \mathsf{NP}^{\Sigma_i^P}$, $\Pi_{i+1}^P = \mathsf{coNP}^{\Sigma_i^P}$. Intuitively, for any complexity class $\mathcal{C}$, a Turing machine with access to a $\mathcal{C}$-oracle can be understood as having a constant-time decision subroutine for problems in $\mathcal{C}$.

For complete problems of the polynomial hierarchy we use satisfiability of quantified Boolean formulas (QBFs). The problem $\text{QBF}_{i,Q}$-TRUTH denotes the problem of deciding satisfiability of a given closed QBF in prenex form, starting with quantifier $Q \in \{\exists, \forall\}$ and i quantifier alternations. For $i \geq 0$ it holds that $\text{QBF}_{i,\exists}$-TRUTH is Σ_i^P-complete and $\text{QBF}_{i,\forall}$-TRUTH is Π_i^P-complete.

As a somewhat non-standard polynomial hierarchy complexity class, we use D_i^P, a generalisation of the complexity class DP to the polynomial hierarchy. A language is in DP iff it is the intersection of a language in NP and a language in coNP. Generally, a language is in D_i^P iff it is the intersection of a language in Σ_i^P and a language in Π_i^P. The canonical problem of $\text{DP} = \text{D}_1^P$ is SAT-UNSAT, the problem to decide for a given pair (ψ_1, ψ_2) of propositional formulas whether ψ_1 is satisfiable and ψ_2 is unsatisfiable. Obviously, by definition $\Sigma_i^P, \Pi_i^P \subseteq \text{D}_i^P \subseteq \Delta_{i+1}^P$ for all $i \geq 0$.

Chapter 3

Defining Semantics via Approximation Fixpoint Theory

The abstract nature of Dung's AFs makes them attractive as a target language for translations from more expressive formalisms. To be more precise, it is common to use expressive languages to model more concrete (argumentation) scenarios, and to provide these original expressive languages with semantics by translating them into Dung AFs (Caminada and Amgoud, 2007; Wyner, Bench-Capon, and Dunne, 2009; Prakken, 2010; Van Gijzel and Prakken, 2011). However, Caminada and Amgoud (2007) observed that it is not always immediately clear how such translations into AFs should be defined, even for a fairly simple source formalism. A major problem that they encountered were unintended conclusions that indirectly led to inconsistency. In the same paper, Caminada and Amgoud also proposed solutions to these problems, where during translation additional precautions have to be taken to avoid undesired anomalies. Let us explain in more detail what this means in general for abstractions among knowledge representation (KR) languages.

First of all, by an abstraction we mean a translation between languages that may disregard some information. Instantiating an abstract language is then the process of translating a more concrete, more expressive language into the abstract, less expressive language. This entails that there is no dichotomy "knowledge representation language vs. abstraction formalism" – any KR language abstracts to a greater or lesser extent, and can thus be used for abstraction purposes. Whether any specific language is to be used for direct, concrete representation or for abstraction of another language depends entirely on the application domain at hand.

Naturally, we are interested in those abstractions that preserve the meaning of translated language elements in some sense. As an example, consider the language $\{yes, no\}$. It is very simple and can abstract from any decision problem whatsoever. Furthermore it is trivial to devise an intuitively correct semantics for it. But to faithfully instantiate this language to a particular decision problem – say, the satisfiability problem of propositional logic –, the problem must be solved during translation, for otherwise the abstraction would not be meaningful at all. At the other end of the spectrum, for any language $\mathcal{L}$, an "abstraction" is provided by $\mathcal{L}$ itself. In contrast to the two-element target language $\{yes, no\}$, using $\mathcal{L}$ as target language makes it trivial to translate $\mathcal{L}$ into the abstraction, but the target language does in fact not abstract at all and devising a semantics for the abstraction is as hard as devising a semantics for the original language.

Thus abstraction proper should indeed disregard some information, but not too much of

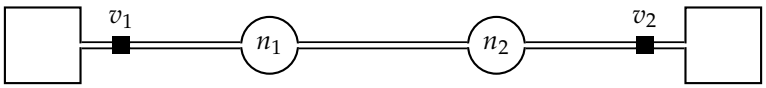
it. In the example above, the fact that the language $\{yes, no\}$ can abstract away from any decision problem is no acceptable argument for its usefulness as an abstraction formalism, since its expressive power is clearly too poor to model real problems (meaning problems that are syntactically different from their solutions). Consequently the expressiveness of a language is important when using it as a target language for abstraction. More specifically, a suitable target language for abstraction must be expressive enough to model important problem aspects, while being sufficiently abstract to ignore irrelevant details.

So to be able to use a formalism for abstraction, we obviously need a clear picture of its capabilities as a KR language, especially its expressive power in comparison to other languages, and about the properties of its semantics. It is one of the main objectives of this chapter to provide this information for abstract dialectical frameworks. For this purpose, we technically view ADFs as KR languages – but of course our work has ramifications for ADFs as abstraction formalisms. In the same way as there is no single intended semantics for argumentation frameworks, there is also no single perfect formalism for abstraction. But to be able to make an informed choice, it is of great importance to understand the inherent relationships between different available options. Our results will facilitate this choice and be an aid to anyone wishing to abstract from concrete argumentation languages; especially, our results will help them decide if they want to translate into AFs or into ADFs.

But why, after all, should there be a choice to be made between AFs and ADFs? Here, the additional expressiveness of ADFs in comparison to AFs comes into play. As we will see throughout this chapter, the well-known distinction between supported and stable models from logic programming is present in ADFs but is missing in AFs. In a different disguise, this same distinction also materialises as Moore expansions vs. Reiter extensions in nonmonotonic logics (Denecker et al., 2003). To summarise it in a nutshell, there are basically two ways in which the major nonmonotonic KR formalisms deal with cyclic positive dependencies between pieces of knowledge. To explain what such cyclic support dependencies are and why they can be problematic, let us look at a study from the literature where researchers applied several logic-based knowledge representation techniques in a medium-sized practical application.

Nogueira, Balduccini, Gelfond, Watson, and Barry (2001) describe a declarative rule-based system that controls some of the functions of a space shuttle. More specifically, the system operates the space shuttle's reaction control system, whose primary responsibility is to manoeuvre the shuttle through space. Part of the rule-based specification represents the plumbing system of this reaction control system. The plumbing system consists of a collection of tanks, jets and pipe junctions, which are connected through pipes. The flow of fluids through pipes is controlled by valves. The purpose of the plumbing system is to deliver fuel and oxidiser from tanks to the jets needed to perform a manoeuvre. The structure of the plumbing system is described by a directed graph whose nodes are tanks, jets and pipe junctions, and whose edges are labelled by valves. The description of the plumbing system should predict how the positions of valves affect the pressure of tanks, jets and junctions. For tanks themselves, the pressure resulting from pressurising certain (other) tanks is easy to specify. For all other nodes in the graph the definition is recursive: roughly, any non-tank node is pressurised by a tank if the node is connected by an open valve to a node which is pressurised by the tank. Nogueira et al. (2001) explicitly recognise that modelling this is non-trivial because the connection graph of the plumbing system can contain cycles. That is, there may be nodes in the graph that are mutually connected to each other, and accurately modelling this is not straightforward:

Example 3.1 (Under Pressure). Consider the following easy setup where two nodes n_1, n_2 with associated tanks are connected to each other. The connection between a node n_i and its tank is controlled by the valve v_i in between.



For the purpose of this example, we assume that the tanks are pressurised. Then obviously, opening v_1 pressurises n_1; likewise, opening v_2 pressurises n_2. But due to the connection in between, it is also the case that pressurising n_1 indirectly pressurises n_2, and pressurising n_2 indirectly pressurises n_1. The easiest way to express all of this in logic programming is via the four rules

$$n_1 \leftarrow v_1 \qquad n_2 \leftarrow v_2$$
$$n_1 \leftarrow n_2 \qquad n_2 \leftarrow n_1$$

where the atoms n_1, n_2 express that the respective node is pressurised, and v_1, v_2 express that the respective valve is open. This way of representing the domain is very elegant in that it is modular: specifying additional parts of the system can be easily achieved by adding new rules – previous rules need not be modified. This is especially important since the real system is going to be considerably more complex.

Now the Clark completion (Clark, 1978) of this program is given by the four propositional formulas $n_1 \leftrightarrow (v_1 \vee n_2)$, $n_2 \leftrightarrow (v_2 \vee n_1)$, $v_1 \leftrightarrow \bot$ and $v_2 \leftrightarrow \bot$. So the valves are considered not open because there are no rules with head v_1 or v_2. The common models of the formulas in the Clark completion lead to the supported model semantics of this program, which considers two states to be possible: $\emptyset$ (where neither of the nodes is pressurised) and $\{n_1, n_2\}$ (where both nodes are pressurised).

But of course, causality dictates that the two nodes cannot simply pressurise each other without an external cause (that is, through an open valve). A reasoner that predicts "both nodes are pressurised" as possible successor state of the state "both nodes are not pressurised" when no relevant valve has been opened in between is obviously not of great assistance – only more so if it offers the cyclic explanation "one node is pressurised because the other is." So the knowledge engineers that specify and use the system should be aware that the supported model semantics does not accurately reflect causality in this domain.

On the other hand, the set $\emptyset$ is the only stable model of the logic program, showing that the stable model semantics correctly deals with the issue at hand. And indeed, Nogueira et al. (2001) explicitly remarked that the ability of answer set programming to express and to reason with recursion allowed them to use a concise definition of pressure. ◇

Such issues with cyclic support dependencies not only occur in logic programs, but also in default logic and autoepistemic logic:

- Cyclic support is allowed by supported semantics for logic programs (which is equivalent to the Clark completion; Clark, 1978) and in expansions of autoepistemic logic (Moore, 1985).
- Cyclic support is disallowed by stable semantics for logic programs (Gelfond and Lifschitz, 1988) and in extensions of default logic (Reiter, 1980).[1]

The fact that this distinction is not present in AFs means that anyone translating their modelling language into AFs has to take care of the issue of cyclic support themselves and thus has to solve part of the problem by hard-wiring it into the translation. (Just like a decision problem

[1]But this is not inherent to these formalisms – both strong expansions for autoepistemic logic that reject cyclic support, and weak extensions for default logic that accept cyclic support can be defined (Denecker et al., 2003).

has to be solved when it is "translated" into the language $\{yes, no\}$.) When ADFs are used as a target language, ADF semantics will simply take care of cyclic supports, thereby considerably simplifying the translation.

Generally speaking, it is at the heart of an abstraction to remove information; it is at the heart of a *good* abstraction to remove *irrelevant* information. If some removed information afterwards turns out to have been relevant, it either has to be (however costly) recomputed or is simply lost. And if the target language cannot natively express some concept, then information about this concept is bound to get lost. An example, again, is the support relation between atoms in a logic program, which is hardly translated into an AF and easily translated into an ADF as this chapter will show.

More concrete empirical evidence for the usefulness of abstract dialectical frameworks has already been provided in the literature. For one, Brewka and Gordon (2010) translated argument evaluation structures of the Carneades framework (Gordon et al., 2007) into ADFs.[2] It is especially remarkable that their work allowed cyclic dependencies among arguments, which was previously not possible in Carneades. Meanwhile, Van Gijzel and Prakken (2011) also translated Carneades into AFs via ASPIC+ (Prakken, 2010). They can deal with cycles, but even with cycles there is only one unique stable, preferred, complete, grounded extension. Thus the semantic richness of abstract argumentation is not used, and more importantly the user cannot choose whether they want to accept or reject positive cyclic dependencies between arguments. In contrast, in the ADF approach of Brewka and Gordon (2010), the user can choose whether support cycles should be accepted or rejected, by choosing models or *stable* models as intended ADF semantics. For another, we show in Chapter 6 how ADFs can be used to provide an argumentation-based semantics for the defeasible theories of Caminada and Amgoud (2007).

To summarise, our main arguments for using abstract dialectical frameworks as abstraction language are the following conclusions of this chapter:

- ADFs are at least as expressive as AFs, and thus can represent all important problem aspects that AFs can represent. On top of that, ADFs offer a built-in treatment of positive cyclic dependencies which is derived from decades of research into nonmonotonic knowledge representation languages.
- ADFs are at most as expressive as normal logic programs, and therefore still sufficiently simple to be suited as an abstraction formalism.
- ADFs provide all of Dung's standard semantics for AFs, so there is no loss in semantical richness. On the contrary, each of the standard AF semantics (stable, preferred, complete, grounded) has at least *two* ADF generalisations.

To go about our main task of analysing the expressiveness of abstract dialectical frameworks, we do not have to start from scratch. Brewka and Woltran (2010) already showed that ADFs are at least as general as AFs and also provided a (non-modular) translation from normal logic programs to ADFs that preserves stable models. However, the exact location of ADFs in the realm of knowledge representation formalisms remained unclear. Later, Brewka, Dunne, and Woltran (2011) were able to give a polynomial translation from ADFs into AFs, suggesting on complexity-theoretical grounds that ADFs are not substantially more expressive than AFs. However, their translation depends on the particular ADF semantics that is used: one does not simply translate ADFs into AFs with a fixed translation and then gets nice correspondences between the ADF and AF semantics (which is exactly how it works the other way

[2] Note that in their approach, an ADF statement corresponds to an argument evaluation structure of Carneades and is hence on the same abstraction level.

around). Rather, to faithfully map ADFs into AFs one has to decide for a semantics beforehand and then apply a semantics-specific translation. Furthermore, the translation introduced by Brewka et al. (2011) for the stable semantics is again not modular, so when something is added to the input ADF, one cannot simply add the translation of the addendum, but has to re-translate the whole updated ADF. In contrast, as we will show, there are translations from AFs and ADFs into normal logic programs (LPs) which are modular, polynomial and faithful with respect to a whole range of semantics.

These and similar results provide us with a more fine-grained view on the location of AFs and ADFs in the bigger picture of existing knowledge representation languages. Technically, we achieve this by a principled and uniform reconstruction of the semantics of abstract dialectical frameworks by embedding them into the approximation operator framework of Denecker, Marek and Truszczyński (henceforth DMT; Denecker et al., 2000, 2003). In seminal work, DMT developed a powerful algebraic framework in which the semantics of logic programs, default logic and autoepistemic logic can be treated in an entirely uniform and purely algebraic way. The approach works by defining operators, and then their fixpoints according to an abstract and principled method. In this chapter, we extend their work by adding abstract dialectical frameworks (and by corollary abstract argumentation frameworks) to their approach.

We do this by defining the so-called *characteristic operator* of an ADF and then deriving new operators following abstract principles (Denecker et al., 2000). For the special case of a Dung argumentation framework, for instance, the characteristic ADF operator fully captures Dung's characteristic function of the AF. Our investigation generalises the most important semantics known from abstract argumentation to the case of ADFs and relates them to the respective logic programming semantics. It will turn out that when generalising AF semantics, there are typically two different possibilities for generalisations: a "supported" and a "stable" version of the respective semantics. Brewka and Woltran (2010) already recognised this in the case of stable extensions for argumentation frameworks: stable AF extensions can be generalised to ADFs in two ways, namely to models and *stable* models for ADFs.

In addition to our usage of operators to clarify the relation of different semantics for *single* formalisms, we will employ another technique to illuminate the relationship between *different* formalisms. This role will be played by investigating polynomial, faithful, modular (PFM) translations between languages as has been done by Gottlob (1995) and Janhunen (1999) for the relationship between nonmonotonic logics. In our case, we even need a stronger kind of translation: "faithful" usually refers to a translation mapping models of one specific semantics of the source formalism to models of another specific semantics for the target formalism. In our case, faithful refers to the translation providing a perfect alignment with respect to *any* fixpoint semantics or at least a range of fixpoint semantics. Of course, this requires all of the involved semantics to be defined for both source and target formalism, which is however the case for our operator-based approach.

The picture that emerges from our work sheds new light on the underlying connections between several classic and novel knowledge representation formalisms, since we study AFs, ADFs and logic programs all in a unified semantical framework. In particular, it conclusively shows that Dung's abstract argumentation frameworks can be seen as special cases of propositional normal logic programs. Now all normal logic programs are default theories, which are in turn theories of autoepistemic logic (Denecker et al., 2003). Thus as a byproduct, our work yields generalisations of argumentation semantics for a general lattice-based setting, from which the existing semantics for logic programming and argumentation can be derived as special cases. Among the semantics generalised are conflict-free and admissible sets, and naive, stage, preferred and semi-stable semantics. As a corollary and another new contribution, this also defines these semantics for default logic and autoepistemic logic (Denecker et al.,

2003). This is a considerable improvement upon a result by Dung (1995), who already argued for a preferred semantics for default logic, but only defined it through a translation to infinite argumentation frameworks. We show that our generalisations of argumentation semantics are well-defined by showing that well-known relationships between the semantics generalise accordingly: for example, any preferred ADF model is also complete.

In the last part of the chapter, we instantiate the general ADF-based operator to the special case of AFs and present new semantical correspondence results between argumentation frameworks and their translated logic programs: preferred and semi-stable extensions correspond one-to-one to M-stable and L-stable models (Saccà and Zaniolo, 1997), respectively. Additionally, we show that our lattice-theoretical account of argumentation yields easier proofs for existing results in this area. As our final result, we prove equivalence (in four-valued Belnap logic) of two different translations from AFs to logic programs: a folklore translation from the literature (we call it the standard translation) that encodes attack by negation as failure, and the original translation of Dung (1995), where attack and defeat of arguments is explicitly recorded.

3.1 Approximate Semantics of ADFs

Abstract dialectical frameworks are knowledge representation formalisms. As such, they allow to express knowledge and provide formal semantics for such expressions. One approach to define semantics for knowledge bases is the one championed by Maarten van Emden, Bob Kowalski and others: there, a revision operator is associated with a knowledge base (Fitting, 2002). The operator revises interpretations for the knowledge base K in the sense that the revision of an interpretation is somehow "more in accord" with the knowledge contained in K. Extending the metaphor, fixpoints of the revision operator then correspond to models since they exactly "hit the spot" in that they represent stationary interpretations that cannot be revised further. In this section, we will apply this operator-based approach to semantics to abstract dialectical frameworks.

From the definition of a model of an ADF by Brewka and Woltran (2010), it is straightforward to devise a two-valued one-step consequence operator for a given ADF: given a two-valued interpretation, we evaluate the acceptance condition of each statement; the resulting evaluation determines the revised interpretation. To generalise this to an approximating operator, we generalise the evaluation from the two-valued $\{\mathbf{t}, \mathbf{f}\}$ to four-valued Belnap logic.

3.1.1 The Characteristic Approximate Operator of an ADF

For an abstract dialectical framework $D = (S, L, C^{\mathbf{t}})$, four-valued interpretations can be represented by pairs (X, Y) with $X, Y \subseteq S$. Such pairs can equivalently be interpreted as approximations to two-valued interpretations where X represents a lower bound and Y an upper bound of the approximation. Given such an approximating pair (X, Y) and an ADF D, to revise the pair we do the following for each statement $s \in S$: we check if there is some subset B of the parents of s (which are exactly the statements that determine the acceptance status of s) such that (1) all statements in B being $\mathbf{t}$ causes s to be $\mathbf{t}$; (2) all statements in B are indeed $\mathbf{t}$ according to the conservative estimate X; (3) the remaining parents of s are indeed $\mathbf{f}$, that is, not contained in the liberal estimate Y. The definition below, the most important definition of this chapter, makes this formally precise.

Definition 3.1. Let $D = (S, L, C^{\mathbf{t}})$ be an abstract dialectical framework. Define an operator $\mathcal{G}_D : 2^S \times 2^S \to 2^S \times 2^S$ (called the *approximate operator* of D) by

$$\mathcal{G}_D(X, Y) = (\mathcal{G}'_D(X, Y), \mathcal{G}'_D(Y, X))$$
$$\mathcal{G}'_D(X, Y) = \left\{ s \in S \;\middle|\; B \in C^{\mathbf{t}}_s, B \subseteq X, (par(s) \setminus B) \cap Y = \emptyset \right\} \qquad \diamond$$

The last condition $(par(s) \setminus B) \cap Y = \emptyset$ can be equivalently reformulated as $par(s) \setminus B \subseteq S \setminus Y$. By $B \subseteq X$ this means that all parents of s which are not **t** must be **f** – there must not be undecided parents of s.

A two-valued immediate consequence operator for ADFs (the equivalent of logic programs' two-valued van Emden-Kowalski operator T_P) is now given by $G_D(X) = \mathcal{G}'_D(X, X)$. The next lemma about this two-valued operator relates to ADF models and will prove useful on various occasions.

Lemma 3.1. *For any abstract dialectical framework $D = (S, L, C)$, statement $s \in S$ and statement set $X \subseteq S$ we have $s \in G_D(X)$ iff $X \cap par(s) \in C^{\mathbf{t}}_s$.*

Proof.

$$\begin{aligned} s \in G_D(X) &\text{ iff } s \in \mathcal{G}'_D(X, X) \\ &\text{iff } X' \in C^{\mathbf{t}}_s, X' \subseteq X, (par(s) \setminus X') \cap X = \emptyset, X \cap par(s) = X' \\ &\text{iff } X \cap par(s) \in C^{\mathbf{t}}_s \end{aligned}$$ □

Our definition of the approximating operator of an ADF immediately defines quite a number of semantics for ADFs, among them all the semantics of Definition 2.2. In the following, we will show how some of the standard operator-based semantics coincide with existing ADF semantics. Operator-based semantics without a corresponding ADF semantics accordingly define new semantical notions for abstract dialectical frameworks, for example three-valued stable models. Similarly, there are ADF semantics that have no operator-based counterpart – BW-stable, BW-admissible and BW-preferred –, we will provide alternative, operator-based definitions for these semantics.

But first, we do the obviously necessary and show that $\mathcal{G}_D$ is indeed an approximating operator. From Definition 3.1 it is immediate that $\mathcal{G}_D$ is symmetric. It is easy to prove that the operator is also $\leq_i$-monotone.

Proposition 3.2. *For any ADF $D = (S, L, C)$, the operator $\mathcal{G}_D$ is $\leq_i$-monotone.*

Proof. Let $(X_1, Y_1) \leq_i (X_2, Y_2)$, that is, $X_1 \subseteq X_2$ and $Y_2 \subseteq Y_1$. We have to show that $\mathcal{G}_D(X_1, Y_1) \leq_i \mathcal{G}_D(X_2, Y_2)$, that is, (1) $\mathcal{G}'_D(X_1, Y_1) \subseteq \mathcal{G}'_D(X_2, Y_2)$ and (2) $\mathcal{G}'_D(Y_2, X_2) \subseteq \mathcal{G}'_D(Y_1, X_1)$.

1. *Let $s \in \mathcal{G}'_D(X_1, Y_1)$. Then there is an $M \in C^{\mathbf{t}}_s$ with $M \subseteq X_1$ and $(par(s) \setminus M) \cap Y_1 = \emptyset$. Now $M \subseteq X_1 \subseteq X_2$; furthermore $Y_2 \subseteq Y_1$ implies $(par(s) \setminus M) \cap Y_2 = \emptyset$, whence $s \in \mathcal{G}'_D(X_2, Y_2)$.*
2. *Analogous.* □

Hence the fixpoints of this operator form a complete sub-lattice of $(2^S \times 2^S, \leq_i)$. From $\mathcal{G}_D$ being approximating it follows that it maps consistent pairs to consistent pairs (Denecker et al., 2000, Proposition 14); in particular its least fixpoint is consistent. Finally, we can construct its associated stable operator $\mathcal{SG}_D$ as defined by Denecker et al. (2000). We will now use our newly defined approximating ADF operator to systematically reconstruct semantical notions for abstract dialectical frameworks.

Conflict-free sets

First of all, we find a nice characterisation of conflict-freeness: a set M is conflict-free for an ADF D iff application of the two-valued immediate consequence operator G_D does not remove elements from M, that is, M is a postfixpoint of G_D. Intuitively speaking, for each statement that is contained in a conflict-free set M, there is no reason not to be contained in M.

Proposition 3.3. *For any abstract dialectical framework $D = (S, L, C)$, a set $M \subseteq S$ is conflict-free for D iff $M \subseteq G_D(M)$.*

Proof.

$$
\begin{aligned}
&M \text{ is conflict-free} \\
&\text{iff for all } s \in M \text{ we have } M \cap par(s) \in C_s^{\mathbf{t}} \\
&\text{iff } M \subseteq \left\{ s \in S \;\middle|\; M \cap par(s) \in C_s^{\mathbf{t}} \right\} \\
&\text{iff } M \subseteq G_D(M) \qquad \text{(by Lemma 3.1)}
\end{aligned}
$$

□

Notice that this characterisation only uses conflict-free *sets* and is thus inherently two-valued. We will later generalise "conflict-free" to three-valued interpretations represented by consistent pairs.

Model semantics

Much in accordance with logic programming, a model of an ADF is simply a two-valued fixpoint of its associated consequence operator:

Proposition 3.4. *For any abstract dialectical framework $D = (S, L, C)$, a set $M \subseteq S$ is a model of D iff $\mathcal{G}_D(M, M) = (M, M)$.*

Proof.

$$
\begin{aligned}
&M \text{ is a model for } D \\
&\text{iff for each } s \in S \text{ we have } s \in M \text{ iff } M \cap par(s) \in C_s^{\mathbf{t}} \\
&\text{iff } M = \left\{ s \in S \;\middle|\; M \cap par(s) \in C_s^{\mathbf{t}} \right\} \\
&\text{iff } M = \mathcal{G}'_D(M, M) \\
&\text{iff } \mathcal{G}_D(M, M) = (M, M)
\end{aligned}
$$

□

Since the correspondence with logic programming is striking, we will use the more specific term "two-valued supported model" from now on.

Stable model semantics

Motivated by the same notion of logic programming, Brewka and Woltran (2010) defined stable models for bipolar ADFs. When we compare their definition to the general operator-based notion of two-valued stable models, we have to acknowledge a slight mismatch.

Example 3.2. Consider the following (bipolar) ADF $\xi = (S, L, C)$ with components $S = \{a, b\}$, $L = \{(a, a), (a, b), (b, b)\}$ and $C_a^{\mathbf{t}} = \{\{a\}\}$ and $C_b^{\mathbf{t}} = \{\emptyset, \{a\}, \{b\}\}$. In words, a supports itself while a and b jointly attack b; in formula notation, $\varphi_a = a$ and $\varphi_b = \neg(a \wedge b)$.

The set $M = \{b\}$ is a model and also a BW-stable model of ξ: The reduct ξ^M is given by the triple $(\{b\}, \emptyset, \{\hat{C}_b^{\mathbf{t}}\})$ with $\hat{C}_b^{\mathbf{t}} = \{\emptyset\}$, an ADF where b is always **t**. (The link (b, b) is not in the reduct because it is attacking in ξ.) However, the operator $\mathcal{G}_\xi$ does not have a two-valued stable model: when trying to reconstruct the upper bound $\{b\}$, we get $\mathcal{G}'_\xi(\emptyset, \{b\}) = \emptyset$ since b attacks itself and thus its containment in the upper bound prevents its inclusion in the new lower bound, as witnessed by $par(b) \cap \{b\} = \{b\} \neq \emptyset$. (Interestingly, this example also shows that M-stable models are not necessarily M-supported: ξ has the single M-stable model $(\emptyset, \{b\})$ and the two M-supported models $(\{a\}, \{a, b\})$ and $(\{b\}, \{b\})$.) ◇

So while there are ADFs with BW-stable models which are not two-valued stable models of the ADF's approximating operator, we can establish an inclusion relation for the converse direction: any operator-based two-valued stable model of an ADF is also a BW-stable model of the ADF. To show this, we first need a lemma that relates the operators $\mathcal{G}'_D(\cdot, M)$ and G_{D^M} whenever M is a model of D.

Lemma 3.5. *Let $D = (S, L, C)$ be a bipolar ADF and (M, M) be a two-valued supported model for D. For any $X \subseteq M$ we find $\mathcal{G}'_D(X, M) \subseteq G_{D^M}(X)$.*

Proof. Recall that the reduct of D with M is defined by $D^M = (M, L^M, C^M)$ with reduced links $L^M = \{(r, s) \mid r, s \in M, (r, s) \in L^+\}$ and for each $s \in M$ and $B \subseteq M$, we have $C_s^M(B) = \mathbf{t}$ iff $C_s(B) = \mathbf{t}$. Now for each $s \in S$ denote by P_s the parent nodes of s with respect to L and for $s \in M$ by P_s^M the parent nodes of s with respect to L^M. It follows that $P_s^M = (M \cap P_s) \setminus \{r \in P_s \mid (r, s) \notin L^+\}$.

Let $s \in \mathcal{G}'_D(X, M)$. (Observe that $X \subseteq M$ means $\mathcal{G}'_D(X, M) \subseteq \mathcal{G}'_D(M, M) = M$ and thus $s \in M$.) Then there is a $B \subseteq P_s$ with $C_s(B) = \mathbf{t}$, $B \subseteq X$ and $(P_s \setminus B) \cap M = \emptyset$. Now $P_s^M \subseteq P_s$ and $X \subseteq M$ yield $(P_s^M \setminus B) \cap X = \emptyset$, whence $X \cap P_s^M \subseteq B$. Define $B' = B \setminus \{r \in P_s \mid (r, s) \notin L^+\}$. By definition $B' \subseteq P_s^M$, whence by $B' \subseteq B \subseteq X$ we get $B' \subseteq X \cap P_s^M$. Since all the removed parents r were attackers (D is bipolar), we still have $C_s(B') = \mathbf{t}$. Now all links from P_s^M to s are supporting and thus still $C_s(X \cap P_s^M) = \mathbf{t}$. Hence $C_s(X \cap P_s^M) = C_s^M(X \cap P_s^M) = \mathbf{t}$ and $s \in G_{D^M}(X)$. □

This shows that G_{D^M} – the two-valued operator associated to the reduced ADF D^M – is in some sense "complete" with respect to the result of $\mathcal{G}'_D(\cdot, M)$ – the operator for checking whether M is a two-valued stable model of D. The next lemma will show that this "completeness" carries over to the least fixpoints of these operators.

Lemma 3.6. *Let $D = (S, L, C)$ be a bipolar ADF and (M, M) be a two-valued supported model for D. If M is the least fixpoint of $\mathcal{G}'_D(\cdot, M)$, then it is the least fixpoint of G_{D^M}.*

Proof. We use the notation from the proof of Lemma 3.5. Let $s \in M$ and observe that we have $C_s(M \cap P_s) = \mathbf{t}$ since M is a model of D. By the definition of the reduct, we get $P_s^M = (M \cap P_s) \setminus \{r \in P_s \mid (r, s) \notin L^+\}$. Since D is bipolar, any link from $(M \cap P_s) \setminus P_s^M$ is attacking and thus $C_s(P_s^M) = \mathbf{t}$.

- *M is a fixpoint of G_{D^M}:*

$$\begin{aligned}
& G_{D^M}(M) \\
&= \left\{ s \in M \;\middle|\; C_s^M(M \cap P_s^M) = \mathbf{t} \right\} && \text{(by definition of } G_{D^M}\text{)} \\
&= \left\{ s \in M \;\middle|\; C_s^M(P_s^M) = \mathbf{t} \right\} && (P_s^M \subseteq M) \\
&= \left\{ s \in M \;\middle|\; C_s(P_s^M) = \mathbf{t} \right\} && \text{(by definition of } C_s^M\text{)} \\
&= \{ s \in M \mid C_s(M \cap P_s) = \mathbf{t} \} && \text{(see above)} \\
&= \{ s \in S \mid C_s(M \cap P_s) = \mathbf{t} \} && (s \in S \setminus M \text{ iff } C_s(M \cap P_s) = \mathbf{f}) \\
&= G_D(M) && \text{(By definition of } G_D\text{)} \\
&= M && (M \text{ is a model of } D)
\end{aligned}$$

- *M is the* least *fixpoint of G_{D^M}: Let $X \subseteq M$ be a fixpoint of G_{D^M}. By Lemma 3.5, $\mathcal{G}'_D(X, M) \subseteq G_{D^M}(X) = X$ and X is a prefixpoint of $\mathcal{G}'_D(\cdot, M)$. Since M is the least fixpoint and thus also the least prefixpoint of $\mathcal{G}'_D(\cdot, M)$, we get $M \subseteq X$.* □

Using the lemma, it is easy to show that the set of BW-stable models contains all operator-based two-valued stable models.

Proposition 3.7. *Let $D = (S, L, C)$ bipolar abstract dialectical framework and $M \subseteq S$. If (M, M) is a two-valued stable model of D, then M is a BW-stable model of D.*

Proof. Let $\mathcal{SG}_D(M, M) = (M, M)$. By definition $M = c\mathcal{G}_D(M) = lfp(\mathcal{G}'_D(\cdot, M))$, that is, M is the least fixpoint of $\mathcal{G}'_D(\cdot, M)$. By Lemma 3.6, M is the least fixpoint of G_{D^M}. Therefore, M is the least model of D^M (and a model of D), thus it is a BW-stable model of D. □

The mismatch noticed in Example 3.2 does not depend on our definition of the four-valued approximating operator: the ADF presented there also does not allow for *ultimate* two-valued stable models, although the model notion of Brewka and Woltran (2010) is perfectly captured by the two-valued one-step ADF consequence operator, which also gives rise to ADF's ultimate family of semantics. Put another way, if we take the model notion from Brewka and Woltran (2010) and apply to it the transformations of Denecker et al. (2004), we arrive at an ultimate stable model semantics which is demonstrably different from BW-stable models.[3]

Thus at the current point, we have two different stable model semantics at our disposal – operator-based two-valued stable models and BW-stable models. The following example shows that the BW-stable semantics admits too many models, since there are ADFs which admit for BW-stable models where one is a proper subset of another.

Example 3.3. Consider the following (bipolar) ADF $\xi = (S, L, C)$ with components $S = \{a, b\}$, $L = \{(a, b), (b, b)\}$ and $C_a^{\mathbf{t}} = \{\emptyset\}$ and $C_b^{\mathbf{t}} = \{\emptyset, \{b\}, \{a, b\}\}$. In words, a is always **t** and attacks b, which however can support itself. The ADF ξ has two BW-stable models, $M_1 = \{a\}$ and $M_2 = \{a, b\}$: The reduct of ξ with M_1 is given by $\xi^{M_1} = (\{a\}, \emptyset, C^{M_1})$ with $C_a^{M_1} = \{\emptyset\}$, thus its least model is $\{a\} = M_1$. For the second BW-stable model $M_2 = \{a, b\}$, the reduct of ξ with M_2 is given by $\xi^{M_2} = (S, \{(b, b)\}, C^{M_2})$ with $C_a^{M_2} = \{\emptyset\}$ and $C_b^{M_2} = \{\emptyset, \{b\}\}$. (Note that the link (b, b) is both supporting and attacking, thus in fact irrelevant.) It is easy to see that $\{a, b\} = M_2$ is the least model of this ADF. In contrast, the approximating operator $\mathcal{G}_\xi$ associated with ξ admits only the single two-valued stable model $(\{a\}, \{a\})$. ◇

[3] We will later do so, see Section 3.3.

The problem with this example is that the ADF ξ allows for the BW-stable model M_2 in which statement b cyclically supports itself. This violates the intuitive requirement of stable semantics that whatever it takes to be true must have a non-cyclic justification. Furthermore, in logic programming, two distinct stable models of normal logic programs cannot be in a subset-relationship; likewise in Reiter's default logic, two distinct extensions of a default theory cannot be in a subset-relationship. With our operator-based definition of two-valued stable models for ADFs, this property comes for free:

Proposition 3.8. *Let $(L, \sqsubseteq)$ be a complete lattice and $\mathcal{O}$ an approximating operator on the bilattice $(L^2, \leq_i)$. For any $x, y \in L$ with $\mathcal{SO}(x,x) = (x,x)$ and $\mathcal{SO}(y,y) = (y,y)$, we have that $x \sqsubseteq y$ implies $x = y$.*

Proof. Let $x, y \in L$ with $\mathcal{SO}(x,x) = (x,x)$, $\mathcal{SO}(y,y) = (y,y)$ and $x \sqsubseteq y$. Since $\mathcal{O}$ is antimonotone in the second component, we have $\mathcal{O}(x,y) \sqsubseteq \mathcal{O}(x,x) = x$ and x is a prefixpoint of $\mathcal{O}(\cdot, y)$. Now y is the least prefixpoint of $\mathcal{O}(\cdot, y)$ and thus $y \sqsubseteq x$. □

Together with Example 3.3, this result means that there is no approximating operator for which Definition 2.1 can reconstruct BW-stable models. However, our operator-based definition of two-valued stable models easily gives rise to an equivalent reduct-based definition of the same concept: in operator terms, M is a two-valued stable model of $\mathcal{G}_D$ iff M is the least fixpoint of the operator $\mathcal{G}'_D(\cdot, M)$. To define a reduct, we have to find the ADF associated to this consequence operator defined for $X \subseteq S$ by

$$\mathcal{G}'_D(X, M) = \left\{ s \in S \;\middle|\; B \in C^{\mathbf{t}}_s, B \subseteq X, (par(s) \setminus B) \cap M = \emptyset \right\}$$

Our new operator-inspired reduct now just has to mimic the way the operator enforces the upper bound M. This is achieved by the definition below, which notably works for all ADFs, bipolar or not.

Definition 3.2. Let $D = (S, L, C^{\mathbf{t}})$ be an abstract dialectical framework. A set $M \subseteq S$ is an *(approximate) stable model* of D iff it is the unique least model of the reduced ADF $D_M = (S, L, C^{\mathbf{t}}_M)$ with

$$B \in C^{\mathbf{t}}_{M,s} \text{ iff } B \in C^{\mathbf{t}}_s, (par(s) \setminus B) \cap M = \emptyset \qquad \diamond$$

Intuitively, the reduct only changes the acceptance functions of statements such that accepting parent configurations that rely on some statement from M being **f** are discarded (since the statements in M are by virtue **t**). If the reduced ADF has a unique least model, and this least model coincides with M, then M is a stable model of the original ADF. It is easy to show that this new reduct-based definition of a stable model coincides with our operator-based definition of two-valued stable models.

Proposition 3.9. *Let $D = (S, L, C^{\mathbf{t}})$ be an abstract dialectical framework and $M \subseteq S$. The pair (M, M) is a two-valued stable model of $\mathcal{G}_D$ iff M is a stable model of D.*

Proof. First observe that we find the two-valued consequence operator of the reduct D_M given for any $X \subseteq S$ by

$$G_{D_M}(X) = \{ s \in S \mid B \in C^{\mathbf{t}}_s, (par(s) \setminus B) \cap M = \emptyset, \; B \subseteq X, (par(s) \setminus B) \cap X = \emptyset \}$$

Hence $X \subseteq M$ implies $G_{D_M}(X) = \mathcal{G}'_D(X, M)$ and the two operators G_{D_M} and $\mathcal{G}'_D(\cdot, M)$ coincide on all subsets of M. In particular, M is the least fixpoint of $\mathcal{G}'_D(\cdot, M)$ iff M is the least fixpoint of G_{D_M}. (The least fixpoint of $\mathcal{G}'_D(\cdot, M)$ always exists since the operator is monotone in $(2^S, \subseteq)$.) Now

$$
\begin{aligned}
&(M, M) \text{ is a two-valued stable model of } \mathcal{G}_D \\
&\text{iff } M \text{ is the least fixpoint of } \mathcal{G}'_D(\cdot, M) \\
&\text{iff } M \text{ is the least fixpoint of } G_{D_M} \\
&\text{iff } M \text{ is the least model of } D_M \\
&\text{iff } M \text{ is a stable model of } D
\end{aligned}
$$

□

Example 3.4. Let us reconsider the problematic ADF from Example 3.3, that is, $D = (S, L, C)$ with components $S = \{a, b\}$, $L = \{(a, b), (b, b)\}$ and $C_a^{\mathbf{t}} = \{\emptyset\}$ and $C_b^{\mathbf{t}} = \{\emptyset, \{b\}, \{a, b\}\}$.

The (new) reduct of D with $M_2 = \{a, b\}$ is given by $D_{M_2} = (S, L, C_{M_2})$ with $C^{\mathbf{t}}_{M_2,a} = \{\emptyset\}$ and $C^{\mathbf{t}}_{M_2,b} = \{\{a, b\}\}$. It is easy to see that $\{a\} \neq M_2$ is the least model of this ADF and M_2 is not a stable model of D.

The (new) reduct of D with $M_1 = \{a\}$ is given by $D_{M_1} = (S, L, C^{\mathbf{t}}_{M_1})$ with $C^{\mathbf{t}}_{M_1,a} = \{\emptyset\}$ and $C^{\mathbf{t}}_{M_1,b} = \{\{a, b\}\}$. Its least model is $\{a\} = M_1$ and M_1 is thus a stable model of D, just as expected. ◇

Admissible sets

For the generalisation of admissibility provided by Brewka and Woltran (2010), the picture is not quite as clear. Firstly, for the special case of Dung argumentation frameworks, any stable extension of an AF is admissible. So we should naturally expect that all ADF generalisations of stable AF extensions are also (the ADF generalisation of) admissible; more specifically, since for AF-based ADFs we have that stable extensions coincide with two-valued supported models of the ADF, for an ADF generalisation of admissibility we should expect that all two-valued supported models of the ADF are also admissible. But this is not the case for the generalisation of admissibility of Brewka and Woltran (2010). Recall that a set M is BW-admissible iff there exists an $R \subseteq S$ such that M is a stable model of $D - R$.

Example 3.5. Consider the simplest abstract dialectical framework with a self-supporting cycle between two arguments, $D = (S, L, C)$ with $S = \{a, b\}$, $L = \{(a, b), (b, a)\}$ and $C_a^{\mathbf{t}} = \{\{b\}\}$, $C_b^{\mathbf{t}} = \{\{a\}\}$. In other words, the links between a and b are both supporting. Hence the set $\{a, b\}$ is a (two-valued supported) model of D, but it is not BW-admissible: $\{a, b\}$ is not a stable model of D or any subframework of D. ◇

It might seem that BW-admissibility is just too restrictive and could be fixed by weakening the definition. One possibility may be to replace "stable" in the definition of BW-admissibility by "supported." But, as the following example shows, already the current, stable-based definition of BW-admissibility considers too many sets to be admissible.

Example 3.6. Consider the (bipolar) ADF $D = (S, L, C)$ with statements $S = \{a, b, c, d\}$, links $L = \{(b, a), (c, a), (d, c)\}$ and acceptance conditions $C_a^{\mathbf{t}} = \{\emptyset, \{b\}, \{c\}\}$, $C_b^{\mathbf{t}} = \{\emptyset\}$, $C_c^{\mathbf{t}} = \{\{d\}\}$ and $C_d^{\mathbf{t}} = \{\emptyset\}$. In words, there is a joint attack of b and c on a – a is **f** if both b and c are **t**, and a is **t** otherwise. Statements b and d are always **t**, and c is **t** if d is. This ADF D has the BW-admissible set $M = \{a, b\}$: Taking $R = \{d\}$, we see that there are no attacks from R to M. Furthermore, the ADF $D - R = \hat{D} = (\hat{S}, \hat{L}, \hat{C})$ is given by $\hat{S} = \{a, b, c\}$, $\hat{L} = \{(b, a), (c, a)\}$ and

$\hat{C}_a = \{\emptyset, \{b\}, \{c\}\}$, $\hat{C}_b = \{\emptyset\}$ and $\hat{C}_c = \{\}$. This ADF $\hat{D}$ has the stable model $\{a, b\}$, which is easily verified when looking at the reduct $\hat{D}^M = (\{a, b\}, \emptyset, \hat{C}^M)$ where $\hat{C}_a^M = \hat{C}_b^M = \{\emptyset\}$. So in a sense, the set $\{a, b\}$ being admissible depends on the removal of $\{d\}$, in which case the only support of c is removed and the joint attack on a cannot happen. But d is by definition of its acceptance condition always **t**, so no reasonable semantics could ever label it **f**, and consequently the condition upon which BW-admissibility of $\{a, b\}$ hinges can never become true.[4] ◇

There is an alternative characterisation of admissibility which satisfies all of our abovementioned criteria. That is, all two-valued supported models of an ADF are admissible in our new sense; and for the ADF from Example 3.6, the undesired BW-admissible set from above is not admissible according to this new definition. As a much more important property, it is defined for *all* ADFs and not only bipolar ones. It is also a generalisation of AF admissibility, as will be shown in Section 3.4.

Intuitively, we require that an admissible pair is first of all consistent and satisfies a further criterion: for any statement that is labelled with either **t** or **f**, the pair must provide sufficient justification for this choice. For a pair (M, N), this means that any statement that is labelled **t** (contained in M) must indeed be accepted by this pair; conversely, any statement that is labelled **f** (not contained in N) must indeed be rejected by the pair. Acceptance and rejection is expressed using the approximating operator, so for (M, N) we require $M \subseteq \mathcal{G}'_D(M, N)$ (justified lower bound) and $\mathcal{G}''_D(M, N) \subseteq N$ (justified upper bound). This combination is easily expressed using the information ordering.

Definition 3.3. For any ADF $D = (S, L, C)$, a consistent pair (M, N) is *admissible* in D iff $(M, N) \leq_i \mathcal{G}_D(M, N)$. ◇

It is clear that the lower bound of an admissible pair (M, N) is a conflict-free set since $M \subseteq \mathcal{G}'_D(M, N) \subseteq \mathcal{G}'_D(M, M) = G_D(M)$. Since for any two-valued supported model M we have $(M, M) = \mathcal{G}_D(M, M)$ it is also immediate that all two-valued supported models of an ADF are (three-valued supported models and in turn) admissible pairs. Interestingly, $\leq_i$-postfixpoints of operators $\mathcal{O}$ were also important for Denecker et al. (2004) – they called them *$\mathcal{O}$-reliable* pairs.

Preferred semantics

In principle, there could be different ways to define the preferred semantics for ADFs: (1) the argumentation way of taking all model candidates that are maximally admissible; (2) the logic-programming way of maximising over three-valued supported models. It is clear that any preferred pair derived according to (2) is also preferred in the sense of (1) since any three-valued supported model is admissible. But – as we will show next – the converse also holds, so it is inessential which of these two definitions we pick. This even holds for any approximating operator on a complete lattice, as is shown by the theorem below; in AF-speak, it expresses the operator generalisation of "all preferred extensions are complete."

Theorem 3.10. *Let $(L, \sqsubseteq)$ be a complete lattice and $\mathcal{O}$ be an approximating operator on $(L^2, \leq_i)$. Any $\leq_i$-maximal admissible pair for $\mathcal{O}$ is a three-valued supported model for $\mathcal{O}$.*

[4] Incidentally, $\{a, b\}$ is also a BW-preferred model that does not contain the BW-well-founded model $\{b, c, d\}$. Since the grounded AF extension is always contained in any preferred AF extension, Example 3.6 also hints at another unexpected (non-)relation between the generalised ADF semantics of Brewka and Woltran (2010).

Proof. Let (x,y) be an $\leq_i$-maximal admissible pair, that is, $(x,y) \leq_i \mathcal{O}(x,y)$ and there is no admissible pair $(\hat{x},\hat{y})$ with $(x,y) <_i (\hat{x},\hat{y})$. We have to show $\mathcal{O}(x,y) = (x,y)$, so assume to the contrary that $\mathcal{O}(x,y) \not\leq_i (x,y)$, that is, $(x,y) <_i \mathcal{O}(x,y)$. Since $\mathcal{O}$ is approximating, it is in particular $\leq_i$-monotone and from $(x,y) \leq_i \mathcal{O}(x,y)$ we can infer $\mathcal{O}(x,y) \leq_i \mathcal{O}(\mathcal{O}(x,y))$. Thus $\mathcal{O}(x,y)$ is itself admissible and $(x,y) <_i \mathcal{O}(x,y)$, in contradiction to (x,y) being $\leq_i$-maximal admissible. □

As an immediate consequence, we have the result that all maximal admissible ADF models are three-valued supported (as we will see, "complete") models.

Corollary 3.11. *Let D be an abstract dialectical framework. Any $\leq_i$-maximal admissible pair is a three-valued supported model.*

This leads to the generalisation of AF preferred semantics for abstract dialectical frameworks (including non-bipolar ones): they are simply M-supported models of $\mathcal{G}_D$, that is, $\leq_i$-maximal fixpoints of $\mathcal{G}_D$. Since supported and stable semantics coincide for argumentation frameworks, another suitable candidate for generalising preferred semantics is the M-stable semantics for ADFs, that is, $\leq_i$-maximal fixpoints of $\mathcal{SG}_D$.

Well-founded semantics In order to generalise the grounded semantics from AFs to ADFs, Brewka and Woltran (2010) introduced – for an ADF $D = (S, L, C)$ – the operator Γ_D on the bilattice $(2^S \times 2^S, \leq_i)$. Motivated by naming conventions from logic programming, they decided to call (the lower bound of) the least fixpoint of Γ_D the "well-founded model" of an ADF. As our next result shows, their intuition of defining the operator was on the spot – they defined the most precise approximation of the two-valued ADF consequence operator G_D.[5]

Lemma 3.12. *For any abstract dialectical framework D, the operator Γ_D is the ultimate approximation of G_D.*

Proof. Recall that for $D = (S, L, C)$ the operator $\Gamma_D : 2^S \times 2^S \to 2^S \times 2^S$ is given by $(X,Y) \mapsto (\Gamma'_D(X,Y), \Gamma''_D(X,Y))$, where

$$\Gamma'_D(X,Y) = \left\{ s \in S \;\middle|\; \text{for all } X \subseteq Z \subseteq Y, \text{we have } Z \cap par(s) \in C_s^{\mathbf{t}} \right\}$$
$$\Gamma''_D(X,Y) = \left\{ s \in S \;\middle|\; \text{there exists } X \subseteq Z \subseteq Y \text{ with } Z \cap par(s) \in C_s^{\mathbf{t}} \right\}$$

Now by (Denecker et al., 2004, Theorem 5.6), for $X \subseteq Y \subseteq S$, the ultimate approximation $\mathcal{U}_D$ of the operator G_D is characterised by $\mathcal{U}_D(X,Y) = (\mathcal{U}'_D(X,Y), \mathcal{U}''_D(X,Y))$ with

$$\mathcal{U}'_D(X,Y) = \bigcap \{ G_D(Z) \mid X \subseteq Z \subseteq Y \}$$
$$\mathcal{U}''_D(X,Y) = \bigcup \{ G_D(Z) \mid X \subseteq Z \subseteq Y \}$$

By Lemma 3.1, we know that for any $s \in S$ and $Z \subseteq S$ we find $Z \cap par(s) \in C_s^{\mathbf{t}}$ iff $s \in G_D(Z)$, which leads to the equalities

$$\begin{aligned}
\mathcal{U}'_D(X,Y) &= \bigcap \{ G_D(Z) \mid X \subseteq Z \subseteq Y \} \\
&= \{ s \in S \mid \text{for all } X \subseteq Z \subseteq Y, \text{we have } s \in G_D(Z) \} \\
&= \left\{ s \in S \;\middle|\; \text{for all } X \subseteq Z \subseteq Y, \text{we have } Z \cap par(s) \in C_s^{\mathbf{t}} \right\} \\
&= \Gamma'_D(X,Y)
\end{aligned}$$

[5] According to personal communication this was conjectured by Mirosław Truszczyński.

and, likewise for the upper bound,

$$\begin{aligned}\mathcal{U}''_D(X,Y) &= \bigcup \{G_D(Z) \mid X \subseteq Z \subseteq Y\} \\ &= \{s \in S \mid \textit{there exists } X \subseteq Z \subseteq Y \textit{ with } s \in G_D(Z)\} \\ &= \left\{s \in S \;\middle|\; \textit{there exists } X \subseteq Z \subseteq Y \textit{ with } Z \cap \textit{par}(s) \in C_s^{\mathbf{t}}\right\} \\ &= \Gamma''_D(X,Y)\end{aligned}$$

which proves the claim. □

This lemma immediately entails that what Brewka and Woltran (2010) called "well-founded" is what DMT call the ultimate Kripke-Kleene semantics.

Corollary 3.13. *For any ADF D, its BW-well-founded semantics coincides with its ultimate Kripke-Kleene semantics.*

The well-founded semantics of D in the usual sense (the least fixpoint of the stable operator $\mathcal{SG}_D$) hence may differ from the BW-well-founded semantics.

Example 3.7 (Continued from Example 2.4). Recall that the ultimate Kripke-Kleene semantics of D is given by the pair $(\{a\}, \{a,b,c,d\})$ (the least model of the operator $\mathcal{U}_D = \Gamma_D$). The well-founded semantics of D in the logic-programming sense is given by the pair $(\{a,d\}, \{a,d\})$. Since this pair is exact, it also represents the unique two-valued stable model of D. (Recall that $M_2 = \{a,d\}$ is the supported model of D where the self-support of b was rejected.) ◇

We have seen how the characteristic operator of an ADF can be used to redefine several existing ADF semantics. The remaining operator-based semantics that we did not yet talk about therefore present new semantics for ADFs. Among them, we generalised complete AF extensions to ADFs (three-valued supported/stable models) which will be explored in more detail in the AF section.

3.2 Relationship to Normal Logic Programs

3.2.1 From ADFs to Logic Programs

We now use the four-valued one-step ADF consequence operator to determine the relationship between ADFs and logic programs. As it turns out, there is a straightforward polynomial and modular translation from ADFs to logic programs which is additionally faithful with respect to all operator-based semantics. The translation creates logic program rules for each statement of a given ADF D. The body of a rule for statement s is satisfied whenever for some $M \subseteq \textit{par}(s)$, the statements in M are **t** and the remaining parents are **f**.

Definition 3.4. Let $D = (S, L, C^{\mathbf{t}})$ be an ADF. Define its *standard logic program* as follows.

$$P(D) = \left\{s \leftarrow (M \cup \textit{not}\,(\textit{par}(s) \setminus M)) \;\middle|\; s \in S, M \in C_s^{\mathbf{t}}\right\}$$ ◇

Example 3.8 (Continued from Example 3.7). The standard logic program $P(D)$ of our running example ADF D is given by

$$a \leftarrow \emptyset \qquad b \leftarrow b \qquad c \leftarrow \{a,b\} \qquad d \leftarrow \textit{not } b$$ ◇

As another illustrative example, we look at the ADF where we observed a mismatch between BW-stable models and operator-based two-valued stable models.

Example 3.9. The ADF D from Example 3.2 is translated into the logic program consisting of the following rules:

$$a \leftarrow a \qquad b \leftarrow \{not\ a, not\ b\} \qquad b \leftarrow \{a, not\ b\} \qquad b \leftarrow \{b, not\ a\}$$

This somewhat obviates why there is no two-valued stable model for D: the only candidate for deriving b in some reduct program is the last rule, which however circularly requires b itself.$\diamond$

The next lemma shows that the term "standard logic program" is well-chosen, since the translation is faithful with respect to all operator-based semantics: the associated approximating operators of an ADF and its standard logic program are identical. The term $\overline{B}$ below denotes the complement of B with respect to the parents of s, that is, $\overline{B} = par(s) \setminus B$.

Lemma 3.14. *For any ADF $D = (S, L, C^{\mathbf{t}})$, we find that $\mathcal{G}_D = \mathcal{T}_{P(D)}$.*

Proof. Let $X, Y \subseteq S$. We show $\mathcal{G}'_D(X,Y) = \mathcal{T}'_{P(D)}(X,Y)$.

$$\begin{aligned}
\mathcal{G}'_D(X,Y) &= \left\{s \in S \;\middle|\; B \in C^{\mathbf{t}}_s, B \subseteq X, \overline{B} \cap Y = \emptyset\right\} \\
&= \left\{s \in S \;\middle|\; s \leftarrow (B \cup not\ \overline{B}) \in P(D), B \subseteq X, \overline{B} \cap Y = \emptyset\right\} \\
&= \left\{s \in S \;\middle|\; s \leftarrow M \in P(D), B = M^+ \subseteq X, \overline{B} = M^-, M^- \cap Y = \emptyset\right\} \\
&= \mathcal{T}'_{P(D)}(X,Y)
\end{aligned}$$

□

This result yields immediate correspondence of all operator-based semantics of an ADF D with the respective semantics of its standard logic program $P(D)$.

Theorem 3.15. *Let $D = (S, L, C^{\mathbf{t}})$ be an abstract dialectical framework and $P(D)$ its standard logic program. Then D and $P(D)$ coincide on all semantics based on their approximation operators.*

In particular, $G_D = T_{P(D)}$ and an ADF and its standard logic program also agree on all semantics derived from the ultimate approximation of their two-valued operators. These results obviate that propositional normal logic programs are at least as expressive as abstract dialectical frameworks in a very strong sense: there exists a *single* translation that preserves models in a whole *type* of semantics. Furthermore, the translation can be computed in polynomial time and is modular with respect to statements.

More precisely, let $D_1 = (S_1, L_1, C^{\mathbf{t}}_1)$ and $D_2 = (S_2, L_2, C^{\mathbf{t}}_2)$ be ADFs such that $S_1 \cap S_2 = \emptyset$. Then the union of the two ADFs is defined as $D_1 \cup D_2 = (S_1 \cup S_2, L_1 \cup L_2, C^{\mathbf{t}}_1 \cup C^{\mathbf{t}}_2)$. For such pairs of ADFs we indeed observe that the translation is modular: $P(D_1 \cup D_2) = P(D_1) \cup P(D_2)$. But it is not straightforward to define the union of two ADFs when they share statements:

Example 3.10. Consider the ADFs $D_1 = (S_1, L_1, C^{\mathbf{t}}_1)$ with $S_1 = \{a, b\}$, $L_1 = \{(b, a)\}$, $C^{\mathbf{t}}_{1,a} = \{\{b\}\}$ and $C^{\mathbf{t}}_{1,b} = \{\emptyset\}$ (in words, b is always **t** and supports a); and $D_2 = (S_2, L_2, C^{\mathbf{t}}_2)$ with $S_2 = \{a, c\}$, $L_2 = \{(c, a)\}$, $C^{\mathbf{t}}_{2,a} = \{\{c\}\}$ and $C^{\mathbf{t}}_{2,c} = \{\emptyset\}$ (in words, c is always **t** and supports a). In both frameworks, the common statement a is supported by a statement which is always **t**. Consequently, a is always **t** for every model of every semantics in both ADFs. However, the union of the acceptance functions' characteristic sets is $C^{\mathbf{t}}_{1,a} \cup C^{\mathbf{t}}_{2,a} = \{\{b\}, \{c\}\}$, and thus in the union ADF, statement a is always **f** since both parents are always **t**. The undesired result in this case is that a is always accepted in the two constituent ADFs but not accepted in their union,

although this union should be expected to exhibit some kind of disjunctive acceptance with respect to its constituents. (For comparison, note that $P(D_1) = \{a \leftarrow b, b \leftarrow \emptyset\}$ and $P(D_2) = \{a \leftarrow c, c \leftarrow \emptyset\}$, whence a is contained in the single (two-valued) stable model $\{a, b, c\}$ of $P(D_1) \cup P(D_2)$.) ◇

Of course, the example above would work if we represented acceptance conditions by formulas $\varphi_{1,a} = b$ and $\varphi_{2,a} = c$: then in the union of the two ADFs the acceptance formula is given by the disjunction $\varphi_{1,a} \vee \varphi_{2,a} = b \vee c$ which has the desired set of models $\{\{b\}, \{c\}, \{b, c\}\}$. However, this is dependent on the specific chosen representation of acceptance conditions, namely propositional formulas. For the general case of overlapping sets of statements and an abstract stance with regard to the representation of acceptance conditions, it seems that a more sophisticated procedure for ADF merging is required. This makes it hard to assess a more general type of modularity concerning translations from ADF into logic programs.

3.2.2 From Logic Programs to ADFs

To translate ADFs into logic programs, we essentially had to take the acceptance formulas, transform them into disjunctive normal form and write an LP rule for each disjunct. To translate logic programs into ADFs, this process is reversed: to devise an acceptance function for statement s, we take the disjunction of all bodies (read as conjunctions of literals) of rules with head s.

Definition 3.5 (Brewka and Woltran, 2010). Let P be a normal logic program over a set A of atoms. Define an ADF $D(P) = (A, L, C^{\mathbf{t}})$ as follows.

- $L = \{(b, a) \mid a \leftarrow M \in P, b \in M^+ \cup M^-\}$
- For $a \in A$, set $C^{\mathbf{t}}_a = \{B \subseteq par(a) \mid a \leftarrow M \in P, M^+ \subseteq B, M^- \cap B = \emptyset\}$. ◇

Alternatively, we could define the acceptance condition of each $a \in A$ by

$$\varphi_a = \bigvee_{a \leftarrow M \in P} \left(\bigwedge_{m \in M^+} m \wedge \bigwedge_{m \in M^-} \neg m \right)$$

Although straightforward, the translation is obviously not modular, since all logic program rules with head a are needed to devise the acceptance condition for statement a. Furthermore, the translation is not faithful with respect to three-valued semantics defined by the approximating operator $\mathcal{G}_D$.

Example 3.11 (Lost in Translation). Consider the following two logic programs over the signature $A = \{a, b, c\}$ that have a common subprogram $P = \{c \leftarrow \emptyset, b \leftarrow not\ b\}$:

1. $P_1 = P \cup \{a \leftarrow b, a \leftarrow c\}$
2. $P_2 = P \cup \{a \leftarrow \{b, not\ c\}, a \leftarrow \{c, not\ b\}, a \leftarrow \{b, c\}\}$

The ADF translations of the two programs are identical: we have $D(P_1) = D(P_2) = (A, L, C^{\mathbf{t}})$ with the obvious links from body atoms to head atoms, $L = \{(b, b), (b, a), (c, a)\}$, statement b being self-attacking, $C^{\mathbf{t}}_b = \{\emptyset\}$, statement c being always $\mathbf{t}$, $C^{\mathbf{t}}_c = \{\emptyset\}$ and statement a being $\mathbf{t}$ if b is $\mathbf{t}$, c is $\mathbf{t}$, or both, $C^{\mathbf{t}}_a = \{\{b\}, \{c\}, \{b, c\}\}$. However, the original logic programs P_1 and P_2 do not have the same three-valued models: While the only (three-valued supported) model of P_1 is $(\{a, c\}, \{a, b, c\})$, the only (three-valued supported) model of P_2 is $(\{c\}, \{a, b, c\})$. That

is, when we force the truth values of c to be true and b to be undefined (in the common subprogram P), the result is that a is true in P_1 (the disjunction $b \vee c$ evaluates to true) but undefined in P_2 (all the conjuncts $b \wedge \neg c$, $c \wedge \neg b$ and $b \wedge c$ evaluate to undefined). $\diamond$

However, the translation is faithful for two-valued supported semantics, as we will show next. Technically, this is proved by establishing a correspondence between the two-valued one-step consequence operators T_P for a logic program P and $G_{D(P)}$ for the logic program's ADF $D(P)$ in the following lemma.

Lemma 3.16. *For any normal logic program P, we have $T_P = G_{D(P)}$.*

Proof. Abbreviate $D(P) = D$, let A be the signature of P and let $X, Y \subseteq A$. We show something slightly more general than $G_D(X) = \mathcal{G}'_D(X, X) = \mathcal{T}'_P(X, X) = T_P(X)$.

1. *$\mathcal{G}'_D(X, Y) \subseteq \mathcal{T}'_P(X, Y)$: Let $a \in \mathcal{G}'_D(X, Y)$. Then there is a $B \in C^{\mathbf{t}}_a$ with $B \subseteq X$ and $\overline{B} \cap Y = \emptyset$. By definition of $D(P)$, there is a $B \subseteq par(a)$ and a rule $a \leftarrow M \in P$ with $M^+ \subseteq B$ and $M^- \cap B = \emptyset$. We have to show that $M^+ \subseteq X$ (this is immediate) and $M^- \cap Y = \emptyset$. Assume to the contrary that there is a $b \in M^- \cap Y$. Then $M^- \cap B = \emptyset$ implies $b \notin B$. Similarly, $\overline{B} \cap Y = \emptyset$ implies that $b \notin \overline{B}$. Thus $b \notin B \cup \overline{B} = par(a)$, which is a contradiction to $b \in M^-$, $a \leftarrow M \in P$ and the definition of $D(P)$.*

2. *$\mathcal{T}'_P(X, X) \subseteq \mathcal{G}'_D(X, X)$: Let $a \in \mathcal{T}'_P(X, X)$. Then there is a rule $a \leftarrow M \in P$ with $M^+ \subseteq X$ and $M^- \cap X = \emptyset$. Define $B = par(a) \cap X$. We have to show that $B \in C^{\mathbf{t}}_a$, $B \subseteq X$ (obvious) and $\overline{B} \cap X = \emptyset$. For the last item, we have that $\overline{B} = par(a) \setminus B = par(a) \setminus (par(a) \cap X) = par(a) \setminus X$, whence $\overline{B} \cap X = \emptyset$. Finally, $a \leftarrow M \in P$ means $M^+ \subseteq par(a)$ and together with $M^+ \subseteq X$ we get $M^+ \subseteq B = par(a) \cap X$. Since $B \subseteq X$, we have $M^- \cap B = \emptyset$. By definition $B \subseteq par(a)$ and thus $B \in C^{\mathbf{t}}_a$.* $\square$

From the proof we can read off that P can derive anything that $D(P)$ can derive, for any three-valued pair; in the converse direction, this only works for two-valued pairs. As an immediate consequence, we get correspondence of two-valued supported models.

Corollary 3.17. *Let P be a normal logic program over a set A of atoms and $D = D(P)$ be its associated abstract dialectical framework. For any set $X \subseteq A$, $\mathcal{G}_D(X, X) = (X, X)$ iff $\mathcal{T}_P(X, X) = (X, X)$.*

As another consequence of the proof of Lemma 3.16, we can also show that LP-based ADFs are sound with respect to two-valued stable models of the LP, that is, any stable model of $D(P)$ is a stable model of P.

Lemma 3.18. *Let P be a normal logic program over a set A of atoms and $D = D(P)$ be its associated abstract dialectical framework. For any set $X \subseteq A$, $\mathcal{SG}_D(X, X) = (X, X)$ implies $\mathcal{ST}_P(X, X) = (X, X)$.*

Proof. Let $\mathcal{SG}_D(X, X) = (X, X)$. Then X is the least fixpoint of $\mathcal{G}'_D(\cdot, X)$ and in particular $\mathcal{G}'_D(X, X) = X$. Now by Lemma 3.16 above, we get $\mathcal{T}'_P(X, X) = X$ and X is a fixpoint of $\mathcal{T}'_P(\cdot, X)$. It remains to show that X is the least fixpoint of $\mathcal{T}'_P(\cdot, X)$. Let Y be a prefixpoint of $\mathcal{T}'_P(\cdot, X)$, that is, $\mathcal{T}'_P(Y, X) \subseteq Y$. By Item 1 in the proof of Lemma 3.16 we have $\mathcal{G}'_D(Y, X) \subseteq \mathcal{T}'_P(Y, X)$, whence $\mathcal{G}'_D(Y, X) \subseteq Y$ and Y is a prefixpoint of $\mathcal{G}'_D(\cdot, X)$. Since X is the least fixpoint of $\mathcal{G}'_D(\cdot, X)$ and thus also its least prefixpoint, we get $X \subseteq Y$ and thus X is the least (pre)fixpoint of $\mathcal{T}'_P(\cdot, X)$. $\square$

The converse of the lemma does not hold:

Example 3.12. Let $P = \{a \leftarrow \emptyset, a \leftarrow a\}$. This program has the two-valued stable model $\{a\}$. Its resulting ADF is $D = D(P) = (\{a\}, \{(a,a)\}, \{C_a^{\mathbf{t}}\})$ with $C_a^{\mathbf{t}} = \{\emptyset, \{a\}\}$. Interestingly, the link (a,a) is both supporting and attacking – that is, it contains no information. When trying to reconstruct the (LP) stable model $\{a\}$, we observe that $\mathcal{G}'_D(\emptyset, \{a\}) = \emptyset$ and $\{a\}$ is not a (ADF) stable model for D. ◇

As much more interesting consequence of Lemma 3.16, it follows that the ultimate approximations of T_P and $G_{D(P)}$ are identical, thus P and $D(P)$ also coincide on all ultimate semantics, including ultimate stable models. This will be the topic of the next section.

By corollary, the above correspondence entails that whatever "goes missing" in the translation from P to $D(P)$ can be recovered by the construction of the ultimate approximation. This should however be taken with a grain of salt, since the ultimate family of semantics are accompanied by higher computational costs (see Chapter 4). So while information thrown away through translation can be recovered, it seems much more economic to keep the information during translation instead of "paying" for a subsequent reconstruction.

3.3 Ultimate Semantics of Abstract Dialectical Frameworks

Now that we have seen how a whole range of argumentation semantics can be defined using a single three-valued operator (the approximate operator of an ADF), we can ask the question why that one operator has to be "the one". It does not. Further families of operator-based ADF semantics are conceivable; one of those is the *ultimate* family. As the name suggests, it applies three-valued operator-based definitions to the ultimate approximation of the two-valued ADF consequence operator. These semantics are the current de-facto standard in the ADF literature.

Definition 3.6. Let $D = (S, L, C)$ be an ADF and (X, Y) be a consistent pair. The interpretation given by (X, Y) is

- *admissible* iff $(X, Y) \leq_i \mathcal{U}_D(X, Y)$;
- *complete* iff $\mathcal{U}_D(X, Y) = (X, Y)$;
- *preferred* iff (X, Y) is $\leq_i$-maximal with respect to being admissible;
- *grounded* iff (X, Y) is the $\leq_i$-least fixpoint of $\mathcal{U}_D$ (its Kripke-Kleene semantics). ◇

As is the case for AFs, also for the ultimate family of semantics for ADFs we have that all preferred models are complete. Moreover, the set of all complete models forms a complete meet-semilattice with the information ordering $\leq_i$ and we can prove the following result, which is a generalisation of Theorem 25 by Dung (1995). The proof makes use of *three-valued interpretations* $v : S \to \{\mathbf{t}, \mathbf{f}, \mathbf{u}\}$, a notational variant of consistent pairs.[6]

Theorem 3.19. *Let D be an ADF.*

1. *Each preferred model is a complete model, but not vice versa.*
2. *The grounded interpretation is the $\leq_i$-least complete model.*
3. *The complete models of D form a complete meet-semilattice with respect to $\leq_i$.*

[6]For a three-valued interpretation $v : S \to \{\mathbf{t}, \mathbf{f}, \mathbf{u}\}$, the associated pair is $(\{s \in S \mid v(s) = \mathbf{t}\}, \{s \in S \mid v(s) \neq \mathbf{f}\})$; conversely, for a consistent pair (X, Y) the associated interpretation maps $X \mapsto \mathbf{t}$, $Y \setminus X \mapsto \mathbf{u}$, and $S \setminus Y \mapsto \mathbf{f}$.

Proof. 1. *If v is preferred, then $v \leq_i \mathcal{U}_D(v)$. We have to show that $\mathcal{U}_D(v) = v$. Assume to the contrary that $\mathcal{U}_D(v) \not\leq_i v$, then $v <_i \mathcal{U}_D(v)$. Since $\mathcal{U}_D$ is $\leq_i$-monotone, we get $\mathcal{U}_D(v) \leq_i \mathcal{U}_D(\mathcal{U}_D(v))$, and $\mathcal{U}_D(v)$ is admissible in contradiction to v being $\leq_i$-maximal admissible. Thus $\mathcal{U}_D(v) \leq_i v$ and v is complete.*

As a counterexample in the opposite direction, consider the ADF in Example 3.14. It has two complete models – its grounded model and the single two-valued model. Only the latter is $\leq_i$-maximal.

2. *The grounded interpretation is the $\leq_i$-least fixpoint of $\mathcal{U}_D$ and thus the $\leq_i$-least complete model.*

3. *Let S be the set of statements in D and define F as the set of all fixpoints of $\mathcal{U}_D$. It is clear that the grounded interpretation of D is the least element of F. Now let $E \subseteq F$ be finite and non-empty. We have to show that E has a greatest lower bound in F. Let e be the greatest lower bound of E in S. The set $\{v : S \to \{\mathbf{t}, \mathbf{f}, \mathbf{u}\} \mid v \leq_i e\}$ forms a complete lattice in which $\mathcal{U}_D$ possesses a greatest fixpoint which is the greatest lower bound of E in F. Now let E additionally be directed. We have to show that it has a least upper bound in F. Let e' be the least upper bound of E in S. The set $\{v : S \to \{\mathbf{t}, \mathbf{f}, \mathbf{u}\} \mid e' \leq_i v\}$ forms a complete meet-semilattice where $\mathcal{U}_D$ possesses a least fixpoint which is the least upper bound of E in F.* □

For the ultimate family of ADF semantics, there also exists a version of the stable model semantics (Brewka, Ellmauthaler, Strass, Wallner, and Woltran, 2013, Definition 6).

Definition 3.7. Let $D = (S, L, C)$ be an ADF with $C = \{\varphi_s\}_{s \in S}$ and $v : S \to \{\mathbf{t}, \mathbf{f}\}$ be a two-valued model of D. Define the reduced ADF D^v with $D^v = (S^v, L^v, C^v)$, where

- $S^v = \{s \in S \mid v(s) = \mathbf{t}\}$
- $L^v = L \cap (S^v \times S^v)$
- $C^v = \{\varphi_s^v\}_{s \in S^v}$ where for each $s \in S^v$, we set $\varphi_s^v = \varphi_s[b/\bot : v(b) = \mathbf{f}]$.

Now the two-valued model v of D is a *stable model* of D if and only if D^v's unique grounded interpretation $grd(D^v)$ satisfies $grd(D^v)(S^v) = \{\mathbf{t}\}$. ◇

In the reduct, in each acceptance formula we replace statements $b \in S$ that v maps to false by their truth value. The rest of the definition straightforwardly expresses the intuition underlying stable models: if all statements the model v takes to be false are indeed false, we must find a constructive proof for all statements the model takes to be true.

Example 3.13. Consider the ADF $D = (\{a, b, c\}, L, C)$ given by

$$\varphi_a = \top, \qquad \varphi_b = \neg a \vee c, \qquad \varphi_c = b$$

In words, a is always accepted, a attacks b, and the links between b and c are support links. According to the original definition of Brewka and Woltran (2010), $\{a, b, c\}$ is the single stable model, violating the basic intuition that all elements of a stable model should have a non-cyclic justification: here b is accepted because c is and vice versa.

It is easy to see that according to our new definition, $M_1 = \{a, b, c\}$ is not stable. The reduced ADF is identical to the original one, and its grounded semantics leaves b and c undefined. On the other hand, $M_2 = \{a\}$ is stable, as intended: the reduced ADF consists of $\varphi_a = \top$ only, and its grounded semantics evaluates a to true. ◇

Example 3.14. Consider the ADF $D = (S, L, C)$ given by

$$\varphi_a = c, \qquad \varphi_b = c, \qquad \varphi_c = a \leftrightarrow b$$

The only two-valued model is $v : S \to \{\mathbf{t}\}$. Since c is true because a and b are and vice versa, the model contains unintended cyclic support and thus should not be stable. Indeed, for the reduct we get $D^v = D$. Let us compute the grounded semantics of D. We start with interpretation $w : S \to \{\mathbf{u}\}$. Since none of the acceptance formulas is a tautology, w is already a fixpoint and thus the grounded interpretation of D. Hence v is not a stable model and D has no stable models, just as intended. Since v is a minimal model of D, the example illustrates that in Definition 3.7 we actually need the grounded semantics; requiring S^v to agree with some $\leq_t$-minimal model of the reduct is insufficient. ◇

The attentive reader may wonder why there is a tailor-made definition of stable semantics although Definition 2.2 defines two-valued stable models also for the ultimate operator, and if the two definitions are different. The main reason for the definition being as it is was pragmatic: Definition 3.7 was prepared for a specific paper (Brewka et al., 2013), in which we did not have the space to introduce approximation fixpoint theory, and so "extracted" the stable model definition.

However, it is not hard to prove that the definition of two-valued stable models given in that paper (Brewka et al., 2013) coincides with Denecker et al.'s ultimate two-valued stable models. We start with an easy observation.

Lemma 3.20. *Let φ be a propositional formula over vocabulary S, and let A, B, C, D be sets with $A \subseteq B \subseteq S$ and $C \subseteq D \subseteq S$.*

$$\left(\varphi^{(A,B)}\right)^{(C,D)} = \varphi^{(A \cup C, B \cap D)}$$

For the actual result (in particular for its proof), it is helpful to recall that the stable models of Brewka et al. (2013) are models by definition.

Proposition 3.21. *The stable model semantics as defined in Definition 3.7 coincides with the ultimate two-valued stable model semantics of Definition 2.2 (Denecker et al., 2004).*

Proof. Let $D = (S, L, C)$ be an ADF and $M \subseteq S$ be a model of D. We show that (M, M) is a Brewka et al.-stable-model of D if and only if (M, M) is an ultimate two-valued stable model of D. First, it is easy to see that M is the lower bound of the ultimate grounded semantics of the reduced ADF $D^M = (M, L \cap (M \times M), C^M)$ if and only if (M, M) is the ultimate grounded semantics of D^M. Furthermore, M is a model of D, whence it is a model of D^M. Thus all acceptance formulas in D^M are satisfiable and for any $X \subseteq M$ we get $\mathcal{U}''_{D^M}(X, M) = M$. That is, during computation of the least fixpoint of $\mathcal{U}_{D^M}$, the upper bound remains constant at M. Now for any $X \subseteq M$ and $s \in S$, we have

$$\begin{aligned}
s \in \mathcal{U}'_D(X, M) \text{ iff } & \varphi_s^{(X,M)} \text{ is a tautology} && \text{(by Definition of } \mathcal{U}'_D\text{)} \\
\text{iff } & \left(\varphi_s^{(\emptyset,M)}\right)^{(X,M)} \text{ is a tautology} && \text{(by Lemma 3.20)} \\
\text{iff } & s \in \mathcal{U}'_{D^M}(X, M) && \text{(by Definition 3.7)}
\end{aligned}$$

So in the complete lattice $(2^M, \subseteq)$, the operators $\mathcal{U}'_D(\cdot, M)$ and $\mathcal{U}'_{D^M}(\cdot, M)$ coincide. Therefore, their least fixpoints coincide. □

We can also show that these stable models are a proper generalisation of Dung's stable extensions.

Theorem 3.22. *Let $F = (A, R)$ be an AF and D_F its associated ADF. For any interpretation v for A, the following are equivalent:*

(A) the set $\{a \in A \mid v(a) = \mathbf{t}\}$ is a stable extension of F,

(B) v is a stable model of D_F,

(C) v is a two-valued model of D_F.

Proof. *Let v be a two-valued interpretation for $D_F = (A, R, C)$ and recall that in D_F, the acceptance formula for each $s \in A$ is $\varphi_s = \bigwedge_{(r,s)\in R} \neg r$.*

"$(A) \Longrightarrow (B)$": Let the set $E = \{a \in A \mid v(a) = \mathbf{t}\}$ be a stable extension of F. Then $E = U_F(E) = \{a \in A \mid b \in E \implies (b, a) \notin R\}$ and $v(a) = \mathbf{t}$ iff for all $b \in E$ the acceptance condition φ_a does not mention b. Additionally, by the definition of the reduct, we have $\varphi_a^v = \varphi_a[b/\bot : b \notin E]$. Hence $\varphi_a^v = \neg\bot \wedge \ldots \wedge \neg\bot \equiv \top$ for all $a \in E$ and in the reduced ADF (E, L^v, C^v) we have that all statements are true in the grounded interpretation. Thus v is stable for D_F, in particular it is a model for D_F and $(B) \Longrightarrow (C)$.

"$(C) \Longrightarrow (A)$": Let v be a model for D_F. Then $a \in A$ means $v(a) = v(\varphi_a) = v\Big(\bigwedge_{(b,a)\in R} \neg b\Big)$. In particular, in the reduct (A^v, L^v, C^v) we have for any $a \in A^v$ that $v(a) = \mathbf{t}$. Thus for all attackers b of a we find $v(b) = \mathbf{f}$ and thus $b \notin A^v$. Conversely, $a \notin A^v$ means $v(a) = \mathbf{f}$ and for some attacker b of a we have $v(b) = \mathbf{t}$ and thus $b \in A^v$. In combination, A^v is a stable extension of F. □

Note that for AF-based ADFs, there is no distinction between models and stable models. The intuitive explanation for this is that stable semantics breaks cyclic supports, which cannot arise in AFs because they cannot express support.

3.4 AF Semantics as Special Cases

In this section we look at the subset of ADFs which corresponds to AFs. Recall that for AFs, the original lattice of interest $(2^A, \subseteq)$ considers sets of arguments and the subset relation. The corresponding bilattice $(2^A \times 2^A, \leq_i)$ is concerned with pairs of sets of arguments and ordered by the information ordering. The elements of this bilattice generalise three-valued approaches to argument evaluation (Verheij, 1996; Caminada, 2006) to the four-valued case: for a pair (S, P), the arguments in $S \cap P$ are *in*, those in $\overline{S \cup P}$ are *out*, those in $P \setminus S$ are *undecided* and those in $S \setminus P$ get the new label *inconsistent*. Consistent pairs (those (S, P) with $S \subseteq P$) can be regarded as three-valued labellings, where exactly all arguments in S are *in*.

As our first observation, we note that the approximating operator that Definition 3.1 assigns to the ADF of an AF F is also a special case of an operator: it is the canonical approximation of U_F, the operator assigning to a set S of arguments all the arguments from A that are not attacked by S.

Proposition 3.23. *For any argumentation framework $F = (A, R)$ and sets $X, Y \subseteq A$, we have $\mathcal{G}_{D(F)}(X, Y) = (U_F(Y), U_F(X))$.*

Proof. *We have to show $\mathcal{G}'_{D(F)}(X, Y) = U_F(Y)$. Recall that $D(F) = (A, R, C^{\mathbf{t}})$, where $C_a^{\mathbf{t}} = \{\emptyset\}$ for*

each $a \in A$. Thus for any argument $a \in A$, we find that $par(a) = R_F^{-1}(a)$. Now

$$\begin{aligned} a \in \mathcal{G}'_{D(F)}(X,Y) \text{ iff } & B \in C_a^{\mathbf{t}}, B \subseteq X, (par(a) \setminus B) \cap Y = \emptyset \\ \text{iff } & B = \emptyset, B \subseteq X, (par(a) \setminus B) \cap Y = \emptyset \\ \text{iff } & par(a) \cap Y = \emptyset \\ \text{iff } & R_F^{-1}(a) \cap Y = \emptyset \\ \text{iff } & a \in U_F(Y) \end{aligned}$$ □

In the remainder, we will denote the four-valued approximation operator of an argumentation framework F by $\mathcal{F}_F$; we formally define $\mathcal{F}'_F = \mathcal{G}'_{D(F)}$. It follows by definition that the characteristic operator $\mathcal{F}_F$ of an AF is its own stable operator:

Lemma 3.24. *For any argumentation framework F, we have $\mathcal{SF}_F = \mathcal{F}_F$.*

Proof. Let $F = (A, R)$ and $X, Y \subseteq A$. We have to show $\mathcal{SF}'_F(X,Y) = \mathcal{F}'_F(X,Y)$. This is easily obtained by considering $\mathcal{SF}'_F(X,Y) = lfp(\mathcal{F}'_F(\cdot, Y)) = lfp(U_F(Y)) = U_F(Y) = \mathcal{F}'_F(X,Y)$. □

This means informally that (in a sense) there are fewer semantics for Dung frameworks than there are for ADFs, logic programming, default logic and autoepistemic logic. Translated into logic programming language, we have that in Dung-style argumentation, supported and stable models coincide, and well-founded semantics equals Kripke-Kleene semantics. Put in different terms of default and autoepistemic logics: for argumentation frameworks, Moore expansions and Reiter extensions coincide!

In principle, this collapsing picture could be due to a mistake in our definition of the characteristic operator. In the following section, it will become clear that this is not the case and the characteristic operator of an argumentation framework is well-designed: we show next how the major semantics of argumentation frameworks can be redefined in terms of fixpoints of the characteristic operator.

3.4.1 Fixpoint Semantics for Abstract Argumentation Frameworks

As a first illustration of universality of the characteristic operator of an AF, we recapitulate a result that is well-known in the argumentation community: the operator U_F (which is at the heart of $\mathcal{F}_F$) can emulate the characteristic function D_F of an argumentation framework: D_F is the same as twofold application of U_F.

Lemma 3.25 (Dung, 1995, Lemma 45). *For any AF F, we have $D_F = U_F^2$.*

For our operator $\mathcal{F}_F$, this means that for any $X, Y \subseteq A$ we have

$$\mathcal{F}_F^2(X,Y) = \mathcal{F}_F(U_F(Y), U_F(X)) = (U_F^2(X), U_F^2(Y)) = (D_F(X), D_F(Y))$$

There are several works in the literature that redefine argumentation semantics in terms of (pre-/post-)fixpoints of the two operators D_F and U_F (Besnard and Doutre, 2004; Grossi, 2012). Since the two operators are closely related and the characteristic approximating operator $\mathcal{F}_F$ can express them both, we can reconstruct argumentation semantics based entirely on this single operator.

We begin with the simplest (in terms of operator characterisation) semantics: recall that for $F = (A, R)$ a set E of arguments is a stable extension iff $E = U_F(E)$, so the following is immediate.

Proposition 3.26. *Let $F = (A, R)$ be an argumentation framework and $E \subseteq A$. Then E is a stable extension of F iff $\mathcal{F}_F(E, E) = (E, E)$.*

It is almost as easy to characterise the class of complete extensions:

Proposition 3.27. *Let $F = (A, R)$ be an argumentation framework and $E \subseteq A$. Then E is a complete extension of F iff for some $E' \subseteq A$ the pair (E, E') is a consistent fixpoint of $\mathcal{F}_F$.*

Proof.

$$
\begin{aligned}
&\text{There is an } E' \subseteq A \text{ with } E \subseteq E' \text{ and } \mathcal{F}_F(E, E') = (E, E') \\
&\text{iff } E \subseteq E' \text{ and } E' = U_F(E) \text{ and } E = U_F(E') \\
&\text{iff } E \subseteq U_F(E) \text{ and } E = U_F^2(E) \\
&\text{iff } E \text{ is conflict-free and } E = D_F(E) \\
&\text{iff } E \text{ is a complete extension.}
\end{aligned}
$$

□

As an easy corollary, we get the grounded semantics as the $\leq_i$-least fixpoint of the characteristic operator. This fixpoint exists since $\mathcal{F}_F$ is $\leq_i$-monotone.

Corollary 3.28. *Let $F = (A, R)$ be an argumentation framework and $E \subseteq A$. Then E is the grounded extension of F iff for some $E' \subseteq A$ the pair (E, E') is the $\leq_i$-least fixpoint of $\mathcal{F}_F$.*

In the sequel, we use the term "complete extension" for the set E and the pair (E, E') interchangeably. It follows by definition that preferred extensions are exactly those consistent fixpoints where E is $\subseteq$-maximal – the M-supported models of $\mathcal{F}_F$.

Proposition 3.29. *Let $F = (A, R)$ be an argumentation framework and $E \subseteq A$. Then E is a preferred extension of F iff for some $E' \subseteq A$ the pair (E, E') is a consistent fixpoint of $\mathcal{F}_F$ where E is $\subseteq$-maximal.*

Alternatively, we can say that for a consistent pair (E, E') the lower bound E is a preferred extension if and only if the pair is M-supported/M-stable for $\mathcal{F}_F$. This immediately yields a "preferred" semantics for default logic, which is an improvement upon a result by Dung (1995, Theorem 43), who defined preferred semantics for default logic only through a translation to infinite AFs.

Semi-stable extensions are those complete ones where the set of arguments in the upper but not in the lower bound (the undecided arguments) is minimal – L-supported/L-stable models.

Proposition 3.30. *Let $F = (A, R)$ be an argumentation framework and $E \subseteq A$. Then E is a semi-stable extension of F iff for some $E' \subseteq A$ the pair (E, E') is a consistent fixpoint of $\mathcal{F}_F$ where $E' \setminus E$ is $\subseteq$-minimal.*

Proof.

$$
\begin{aligned}
&E \cup R_F(E) \text{ is } \subseteq\text{-maximal} \\
&\text{iff } E \cup \overline{U_F(E)} \text{ is } \subseteq\text{-maximal} \\
&\text{iff } E \cup \overline{E'} \text{ is } \subseteq\text{-maximal} \\
&\text{iff } \overline{E \cup \overline{E'}} \text{ is } \subseteq\text{-minimal} \\
&\text{iff } \overline{E} \cap E' \text{ is } \subseteq\text{-minimal} \\
&\text{iff } E' \setminus E \text{ is } \subseteq\text{-minimal}
\end{aligned}
$$

□

Finally, we show that the ADF version of "admissible" (Definition 3.3) is a proper generalisation of the respective AF notion. This is easily shown using the respective associated approximating operators.

Proposition 3.31. *Let $F = (A, R)$ be an argumentation framework and $X \subseteq A$. Then X is an admissible set for F iff $(X, U_F(X))$ is an admissible pair for $\mathcal{F}_F$.*

Proof. Abbreviate $Y = U_F(X)$. We have the following equivalences:

$$
\begin{aligned}
&X \text{ is an admissible set for } F\\
&\text{iff } X \text{ is conflict-free and } X \subseteq D_F(X)\\
&\text{iff } X \subseteq U_F(X) \text{ and } X \subseteq U_F^2(X)\\
&\text{iff } X \subseteq Y \text{ and } X \subseteq U_F(Y) \text{ and } U_F(X) \subseteq Y\\
&\text{iff } X \subseteq Y \text{ and } (X, Y) \leq_i (U_F(Y), U_F(X))\\
&\text{iff } (X, Y) \text{ is consistent and } (X, Y) \leq_i \mathcal{F}_F(X, Y)\\
&\text{iff } (X, Y) \text{ is an admissible pair for } \mathcal{F}_F
\end{aligned}
$$

□

Jakobovits and Vermeir (1999) introduced four-valued labellings for argumentation frameworks. Using indicators for acceptance ($+$) and rejection ($-$), they define the labels $\mathbf{t} = \{+\}$, $\mathbf{f} = \{-\}$, $\mathbf{u} = \{+, -\}$ and *irrelevant* $= \emptyset$. It is possible to adapt our intuition behind pairs (X, Y) of sets of arguments in the sense that those in $X \setminus Y$ are irrelevant (instead of inconsistent); in this case, their labels can be seen as indicating which of the two possible statuses $+$ and $-$ are still considered possible for the argument in question. Under this assumption, it is straightforward to adapt the definitions of (Jakobovits and Vermeir, 1999) to our setting. To this end, we first recall their original definition of four-valued labellings.

Definition 3.8 (Jakobovits and Vermeir, 1999, Definition 3.1). Let $F = (A, R)$ be an AF. A *JV-labelling* is a function $l : A \to 2^{\{+,-\}}$ such that for all $a \in A$:

1. If $- \in l(a)$, then there is a $b \in A$ with $(b, a) \in R$ and $+ \in l(b)$.
2. If $+ \in l(a)$, then $(b, a) \in R$ implies $- \in l(b)$.
3. If $+ \in l(a)$, then $(a, c) \in R$ implies $- \in l(c)$.

A JV-labelling is *total* iff $l(a) \neq \emptyset$ for all $a \in A$.[7] ◇

It is easy to establish a correspondence between pairs (X, Y) and functions $l : A \to 2^{\{+,-\}}$. It only remains to reformulate the three conditions in terms of AF operators, which results in the proposition below.

Proposition 3.32. *Let $F = (A, R)$ be an AF and $X, Y \subseteq A$.*

(A) *The pair (X, Y) corresponds to a JV-labelling iff $X = \mathcal{F}_F'(X, Y)$ and $Y \subseteq \mathcal{F}_F''(X, Y)$.*

(B) *The pair (X, Y) is consistent iff its corresponding JV-labelling is total.*

Proof. (A) *Define $l : A \to 2^{\{+,-\}}$ such that it maps as follows: $X \cap Y \mapsto \{+\}$, $A \setminus (X \cup Y) \mapsto \{-\}$, $Y \setminus X \mapsto \{+, -\}$ and $X \setminus Y \mapsto \emptyset$. We first observe that for any argument $a \in A$, we have $- \in l(a)$ iff $a \notin X$, and $+ \in l(a)$ iff $a \in Y$. The conditions of Definition 3.8 are now readily formulated thus:*

[7] Jakobovits and Vermeir (1999) call such labellings complete, which we however use with a different meaning.

1. *If $a \notin X$, then a is attacked by Y; that is, $U_F(Y) \subseteq X$.*
2. *If $a \in Y$, then $(b,a) \in R$ implies $b \notin X$; that is, $Y \subseteq U_F(X)$.*
3. *If $a \in Y$, then $(a,c) \in R$ implies $c \notin X$; that is, $X \subseteq U_F(Y)$.*

Proposition 3.23 now yields the claim.

(B) *(X,Y) is consistent iff $X \subseteq Y$ iff $X \setminus Y = \emptyset$ iff $\{a \in A \mid l(a) = \emptyset\} = \emptyset$ iff l is total.* □

Using the truth and information orderings, pairs (X,Y) that correspond to JV-labellings could be characterised by $(X,Y) \leq_t \mathcal{F}_F(X,Y)$ and $\mathcal{F}_F(X,Y) \leq_i (X,Y)$. While this may suggest a possible intuitive reading of such pairs, we have to be cautious as the intuitions underlying these orderings are slightly different.

We have seen how all of the semantical notions for AFs considered in this chapter can be recast in terms of the approximating operator of an AF, as fixpoints or pre-/postfixpoints with respect to the information ordering $\leq_i$ and truth ordering $\leq_t$. This tells us that operators associated with an argumentation framework are useful tools to study the semantics of the AF. This technique of associating operators with a knowledge base and then studying the operators to study the knowledge base is successfully and widely used in logic programming. In the next section, we will see that this enables us to elegantly build a bridge from abstract argumentation to logic programming via the approximation operators associated with the respective objects of study.

3.4.2 From Argumentation Frameworks to Logic Programs

There are different translations from AFs into LPs in the literature: the one we call the standard translation, and the one devised by Dung (1995) to demonstrate how logic programs could be used to implement abstract argumentation. We consider each of the translations in turn and lastly show that they produce equivalent logic programs.

Standard Translation

The translation we refine below seems to have first appeared in a paper by Osorio, Zepeda, Nieves, and Cortés (2005), but is also referred to as "well-known" by Gabbay and d'Avila Garcez (2009, Example 1.2). Those authors do not provide a definition or motivation for the translation, but our subsequent results will show that the intuition behind it is sound and the name "standard translation" is justified.

The standard logic program resulting from an AF uses the set of arguments as its underlying signature. A rule is created for each argument a, and the rule basically says "a is accepted if none of its attackers is accepted."

Since AFs are in particular ADFs, the standard logic program of an AF F is given by $P(D(F))$, that is, translating the AF F into an ADF $D(F)$ and that further into the standard LP of the ADF. For AFs $F = (A,R)$, the definition of its standard logic program can be simplified to the following:

$$P(F) = \{a \leftarrow \{not\ b \mid (b,a) \in R\} \mid a \in A\}$$

Note that the positive body is empty in general since there is no notion of support in classical Dung-style AFs. Also, the negative bodies of the rules are finite if and only if the framework is finitary.

It should be noted that the standard translation from AFs to LPs is not modular, since the LP rule for an atom a depends on all attackers of a. This might seem paradoxical at first, since the standard translation from ADFs to LPs *is* modular with respect to statements. But recall that the union of two ADFs is defined whenever the two have disjoint statements, so for AFs with disjoint sets of arguments the standard translation is again modular with respect to arguments.

It is immediate from Lemma 3.14 that the associated operators of AFs F and their translated logic program $P(F)$ are the same.

Corollary 3.33. *For any argumentation framework F, we have $\mathcal{F}_F = \mathcal{T}_{P(F)}$.*

Now we know from Lemma 3.24 that the approximation operator of any AF F is its own stable operator – in symbols $\mathcal{F}_F = \mathcal{SF}_F$. Combining these two results about $\mathcal{F}_F$ leads to the following lemma, which nicely pictures the special role of argumentation frameworks in the realm of nonmonotonic reasoning formalisms.

Lemma 3.34. *For any AF F, we have $\mathcal{T}_{P(F)} = \mathcal{F}_F = \mathcal{SF}_F = \mathcal{ST}_{P(F)}$.*

Since the consequence operator of a logic program yields its Kripke-Kleene and well-founded models as well as its two-valued and three-valued supported and stable models, this lemma immediately gives rise to several important coincidence results, accumulated in the first main result of this section below. Its first and last items are obvious. The second item contains the conclusion of Wu, Caminada, and Gabbay (2009, Theorem 17) (they did not look at supported semantics), while the third and fourth items imply new results that solve open problems posed there.

Theorem 3.35. *Let F be an AF. The following are identical:*

1. *the grounded extension of F, the Kripke-Kleene model of $P(F)$ and the well-founded model of $P(F)$;*
2. *complete extensions of F, three-valued supported models of $P(F)$ and three-valued stable models of $P(F)$;*
3. *preferred extensions of F, M-supported models of $P(F)$ and M-stable models of $P(F)$;*
4. *semi-stable extensions of F, L-supported models of $P(F)$ and L-stable models of $P(F)$;*
5. *stable extensions of F, two-valued supported models of $P(F)$ and two-valued stable models of $P(F)$.*

Proof. The first item is obvious, since they are the least fixpoint of the same operator; the rest follows from Lemma 3.34 and Propositions 3.26, 3.27, 3.29 and 3.30. □

As witnessed by Lemma 3.14, for the standard translation the correspondence between AFs and LPs is immediate. We will next consider a different translation where this correspondence is less obvious, albeit still present. Most importantly, that translation will be modular for all argumentation frameworks.

Dung's Translation

Dung duplicates the arguments, thereby explicitly keeping track of their being *in* or *out*: for $a \in A$, a new propositional variable *-a* expresses defeat of a by some counterargument. Note

that this translation is modular with respect to both arguments and attacks, and furthermore rule bodies are always finite.[8]

Definition 3.9. Let $F = (A, R)$ be an argumentation framework. Define $\text{-}A = \{\text{-}a \mid a \in A\}$, $A^{\pm} = A \cup \text{-}A$ and a logic program over $A^{\pm}$ as follows.

$$P_D(F) = \{a \leftarrow \mathit{not}\ \text{-}a \mid a \in A\} \cup \{\text{-}a \leftarrow b \mid (b, a) \in R\} \qquad \diamond$$

Intuitively, an argument a is accepted (signified by atom a) unless it is defeated (signified by atom $\text{-}a$). An argument is defeated if it is attacked by an accepted argument.

We show next that the four-valued one-step consequence operator for the logic program resulting from Dung's translation essentially does the same as the characteristic operator of the original argumentation framework. It only needs twice as many steps due to the syntactical duplication of arguments.

To show this result, we need the technical notion of coherence: in words, a pair is coherent if it respects the intuition of $\text{-}a$ for $a \in A$, in the sense that a is true iff $\text{-}a$ is false and vice versa. A pair (S, P) of sets of arguments can be extended to matching pairs (S^*, P^*) of logic program atoms over $A^{\pm}$ in a straightforward way.

Definition 3.10. Let A be a set of arguments and $S^*, P^* \subseteq A^{\pm}$. The pair (S^*, P^*) is *coherent* iff for all $a \in A$, we find $a \in S^*$ iff $\text{-}a \notin P^*$ and $a \in P^*$ iff $\text{-}a \notin S^*$. For $S, P \subseteq A$, define $co(S, P) = (S \cup \text{-}\overline{P}, P \cup \text{-}\overline{S})$.[9] $\diamond$

Observe that $\text{-}\overline{X} = \{\text{-}a \mid a \notin X\}$, so it is clear that the pair $co(S, P)$ is coherent. What the function does, intuitively, is simple: if a is not in the upper bound P, that is, cannot become true any more, then it can be considered false, which is expressed by adding $\text{-}a$ to the lower bound; likewise, if a is not in the lower bound S, that is, is not yet considered true, then its falsity must be considered an option, which leads to $\text{-}a$ being added to the upper bound. These manipulations are entirely syntactic and do not mention attacks.

We are now ready to show that for an AF $F = (A, R)$, its standard translation $P(F)$ and Dung translation $P_D(F)$ have the same four-valued supported models with respect to the original signature A. Technically, we show that the fixpoints of their four-valued one-step consequence operators coincide.

Theorem 3.36. *Let $F = (A, R)$ be an argumentation framework with standard translation P and Dung translation P_D and let $S, P \subseteq A$.*

$$\mathcal{T}_P(S, P) = (S, P) \quad \textit{iff} \quad \mathcal{T}_{P_D}(co(S, P)) = co(S, P)$$

Proof. We first observe that for any $X, Y \subseteq A$ and $a \in A$, by definition of P_D we have $a \in \mathcal{T}'_{P_D}(X, Y)$ iff $\text{-}a \notin Y$ and $\text{-}a \in \mathcal{T}'_{P_D}(X, Y)$ iff $a \notin U_F(X)$, whence $\mathcal{T}'_{P_D}(X, Y) = \{a \mid \text{-}a \notin Y\} \cup \text{-}\overline{U_F(X)}$ and

[8] Dung's original translation is slightly different; he uses a first-order signature and logic program atoms with variables (Dung, 1995). Definition 3.9 above is merely a syntactical variant that already incorporates ground instantiation.

[9] The notation is entirely unambiguous since for any $S \subseteq A$ we have $\text{-}\overline{S} = \overline{\text{-}S}$.

$\mathcal{T}'_{P_D}(X, \text{-}\overline{Y}) = Y \cup \text{-}\overline{U_F(X)}$. *Now*

$\mathcal{T}_P(S,P) = (S,P)$
iff $\mathcal{F}_F(S,P) = (S,P)$
iff $(U_F(P), U_F(S)) = (S,P)$
iff $S = U_F(P)$ *and* $P = U_F(S)$
iff $S = U_F(P)$ *and* $P = U_F(S)$ *and* $\text{-}\overline{U_F(P)} = \text{-}\overline{S}$ *and* $\text{-}\overline{U_F(S)} = \text{-}\overline{P}$
iff $S \cup \text{-}\overline{U_F(S)} = S \cup \text{-}\overline{P}$ *and* $P \cup \text{-}\overline{U_F(P)} = P \cup \text{-}\overline{S}$
iff $(S \cup \text{-}\overline{U_F(S)}, P \cup \text{-}\overline{U_F(P)}) = (S \cup \text{-}\overline{P}, P \cup \text{-}\overline{S})$
iff $(\mathcal{T}'_{P_D}(S, \text{-}\overline{S}), \mathcal{T}'_{P_D}(P, \text{-}\overline{P})) = (S \cup \text{-}\overline{P}, P \cup \text{-}\overline{S})$
iff $(\mathcal{T}'_{P_D}(S \cup \text{-}\overline{P}, P \cup \text{-}\overline{S}), \mathcal{T}'_{P_D}(P \cup \text{-}\overline{S}, S \cup \text{-}\overline{P})) = (S \cup \text{-}\overline{P}, P \cup \text{-}\overline{S})$
iff $\mathcal{T}_{P_D}(S \cup \text{-}\overline{P}, P \cup \text{-}\overline{S}) = (S \cup \text{-}\overline{P}, P \cup \text{-}\overline{S})$
iff $\mathcal{T}_{P_D}(co(S,P)) = co(S,P)$ □

Furthermore, coherent pairs are also the *only* fixpoints of $\mathcal{T}_{P_D}$.

Proposition 3.37. *Let $F = (A, R)$ be an AF, P_D be its Dung translation over $A^{\pm}$ and let $S^*, P^* \subseteq A^{\pm}$. If $\mathcal{T}_{P_D}(S^*, P^*) = (S^*, P^*)$, then (S^*, P^*) is coherent.*

Proof. *Let $\mathcal{T}_{P_D}(S^*, P^*) = (S^*, P^*)$. Thus by the proof of Theorem 3.36 above, it is the case that both $S^* = \{a \mid \text{-}a \notin P^*\} \cup \text{-}\overline{U_F(S^*)}$ and $P^* = \{a \mid \text{-}a \notin S^*\} \cup \text{-}\overline{U_F(P^*)}$. For $a \in A$, it immediately follows that $a \in S^*$ iff $\text{-}a \notin P^*$ and $a \in P^*$ iff $\text{-}a \notin S^*$, thus (S^*, P^*) is coherent.* □

Hence for any semantics derived from the operator $\mathcal{T}_{P_D}$ which is only "interested" in atoms from A, the choice between standard translation and Dung translation is semantically inessential. We remark that Dung's translation has the advantage of producing a logic program where each rule has a finite body.

Theorem 3.36 and Proposition 3.37 immediately yield the same nice correspondence picture from the standard translation (Theorem 3.35) for Dung's translation. The first and last items below are again obvious for our setting, parts of them were also proved by Dung (1995, Theorem 62). Correspondence results 2, 3 and 4 are completely new.

Theorem 3.38. *Let $F = (A, R)$ be an argumentation framework. The following are in one-to-one correspondence:*

1. *the grounded extension of F, the Kripke-Kleene model of $P_D(F)$ and the well-founded model of $P_D(F)$;*
2. *complete extensions of F, three-valued supported models of $P_D(F)$ and three-valued stable models of $P_D(F)$;*
3. *preferred extensions of F, M-supported models of $P_D(F)$ and M-stable models of $P_D(F)$;*
4. *semi-stable extensions of F, L-supported models of $P_D(F)$ and L-stable models of $P_D(F)$;*
5. *stable extensions of F, two-valued supported models of $P_D(F)$ and two-valued stable models of $P_D(F)$.*

Proof. *Follows from Theorem 3.36, Proposition 3.37 and Propositions 3.26, 3.27, 3.29 and 3.30.* □

This theorem conclusively shows that Dung's modular translation from AFs to LPs is faithful with respect to all operator-based semantics. We infer that propositional normal logic programs are at least as expressive as abstract argumentation frameworks.

3.4.3 From Logic Programs to Argumentation Frameworks

For ADFs, we have seen how the standard translation into logic programs could straightforwardly be reversed into a translation from normal logic programs to ADFs that was sound with respect to both two-valued supported and stable model semantics. In the case of AFs, however, things are different.

Dung (1995) defined two semantics-independent translations from normal logic programs into argumentation frameworks. When restricted to propositional programs, his first translation (Section 4.3.2) is polynomial and faithful with respect to two-valued supported models and the Kripke-Kleene semantics, but not modular. Furthermore it is not faithful with respect to two-valued stable models: the logic program $\{a \leftarrow a\}$ has the only two-valued stable model $\emptyset$, but its associated argumentation framework[10] $(\{a, \neg a\}, \{(a, \neg a), (\neg a, a)\})$ has two stable extensions (corresponding to the logic program's two-valued supported models). For Dung's second translation (Section 4.3.1), the size of the resulting argumentation framework may – in the worst case – be at least exponential in the number of atoms in the vocabulary of the logic program.

Although it is certainly possible to devise polynomial, semantics-dependent translations from logic programs into argumentation frameworks (as a start, consider translating a logic program into an ADF to which in turn the translation from Brewka et al. (2011) is applied), we consider it unlikely that any such translation is polynomial, faithful *and* modular. In particular, it is highly unlikely that a polynomial and modular translation is faithful with respect to both supported *and* stable semantics, as these two semantics are not equal in general but coincide for abstract argumentation frameworks.

3.5 General Semantics for Approximating Operators

We have seen how the characteristic operator of an ADF can be used to redefine the existing ADF semantics. In addition, this introduced the admissible, preferred and stable semantics for all ADFs – they were previously only defined for bipolar ADFs. We have also seen that an ADF D and its standard logic program $P(D)$ correspond on all semantics which are defined for both ADFs and LPs. Finally, we have seen how the characteristic operator of Dung-style argumentation frameworks (given by AF-based ADFs) allows to redefine AF semantics for operators. This allows us to easily transfer definitions of semantics from abstract argumentation to abstract dialectical frameworks, logic programming and beyond – to the general case of approximating operators.

3.5.1 Admissible

In Dung argumentation frameworks, a set of arguments is admissible if it is conflict-free and defends itself against all attacks. For abstract dialectical frameworks, we have seen in Definition 3.3 and Proposition 3.31 that a suitable ADF generalisation of AF admissibility is given by consistent pairs that are postfixpoints with respect to the information ordering $\leq_i$. These pairs have the property that applying the revision operator increases (or preserves) their information content. For the sake of completeness we have included the following formal definition.

[10] Actually, applying the translation yields an argumentation framework that looks slightly more complicated: $(\{(\{a\}, a), (\{\neg a\}, \neg a)\}, \{(((\{a\}, a), (\{\neg a\}, \neg a)), ((\{\neg a\}, \neg a), (\{a\}, a)))\})$. We chose to simplify notation for the sake of readability.

Definition 3.11. Let $(L, \sqsubseteq)$ be a complete lattice and $\mathcal{O}$ an approximating operator on the bilattice $(L^2, \leq_i)$. A consistent pair $(x, y) \in L^2$ is *admissible for* $\mathcal{O}$ iff $(x, y) \leq_i \mathcal{O}(x, y)$. $\diamond$

Denecker et al. (2004) already took special note of admissible pairs and called them $\mathcal{O}$-reliable. They point out that $\mathcal{O}$-reliable pairs – consistent pairs whose $\mathcal{O}$-revisions are at least as accurate – are especially useful for studying fixpoints of O, the original operator that $\mathcal{O}$ approximates. In particular, the $\leq_i$-least element $(\bot, \top)$ is $\mathcal{O}$-reliable; iterating $\mathcal{O}$ on it leads to the Kripke-Kleene semantics, which provides a more precise approximation to all fixpoints of the approximated operator O.

3.5.2 Semi-stable

Theorem 3.35 and Proposition 3.30 immediately yield a definition of L-stable/semi-stable semantics for default and autoepistemic logics. Complete semantics for the two are given by consistent fixpoints (those (x, y) with $x \sqsubseteq y$) of an approximating operator. To generalise semi-stable to operators we simply have to generalise the minimality criterion of L-stable models for logic programming. Since this involves algebraic operations on lattice elements, we have to make some more restricting assumptions on the underlying lattice.

In the following definition, for a complete lattice $(L, \sqsubseteq)$ with join $\sqcup$ and meet $\sqcap$, we assume the existence of a function $\cdot^{-1} : L \to L$ such that for any $x, y \in L$,

- $(x^{-1})^{-1} = x$ ($\cdot^{-1}$ is involutive)
- $(x \sqcup y)^{-1} = x^{-1} \sqcap y^{-1}$ and $(x \sqcap y)^{-1} = x^{-1} \sqcup y^{-1}$ (de Morgan's laws)

In the special cases we have seen so far, the role of this "negation" is played by set complement with respect to the underlying vocabulary.

Definition 3.12. Let $(L, \sqsubseteq)$ be a complete lattice and $\mathcal{O}$ an approximating operator on its corresponding bilattice $(L^2, \leq_i)$. A consistent pair (x, y) is *L-supported* iff it is a fixpoint of $\mathcal{O}$ and $y \sqcap x^{-1}$ is $\sqsubseteq$-minimal. A consistent pair (x, y) is *L-stable* iff it is a fixpoint of $\mathcal{SO}$ and $y \sqcap x^{-1}$ is $\sqsubseteq$-minimal. $\diamond$

For the special case of argumentation, these general definitions of L-supported and L-stable reduce to a consistent fixpoint (S, P) of $\mathcal{F}_F = \mathcal{SF}_F$ such that $P \cap \overline{S} = P \setminus S$ (the set of undecided arguments) is $\subseteq$-minimal – a semi-stable extension.

3.5.3 Conflict-free (Asymmetric)

In classical abstract argumentation, a set of arguments is conflict-free if there are no attacks amongst its members. For abstract dialectical frameworks, a set of statements is conflict-free if each statement – informally speaking – has no reason not to be in the set. Such a reason could be the presence of an attackers as well as the absence of supporters in case the statement's acceptance conditions so requires.

To generalise this notion to three-valued pairs (X, Y), we require two things:

- any **t** statement (in X) must have a reason not to be **f**, and
- any **f** statement (not in Y) must have a reason to be **f**.

Notice the asymmetry, which resurfaces in the following operator-based definition. For a consistent pair to be conflict-free, we stipulate that applying the approximating operator improves the upper bound of the pair.

Definition 3.13. Let $(L, \sqsubseteq)$ be a complete lattice and $\mathcal{O}$ an approximating operator on the bilattice $(L^2, \leq_i)$. A consistent pair $(x, y) \in L^2$ is *asymmetric conflict-free for* $\mathcal{O}$ iff $x \sqsubseteq \mathcal{O}''(x, y) \sqsubseteq y$.$\diamond$

Let us illustrate this notion on a standard example AF. It shows that asymmetric conflict-free pairs allow to set arguments to **u** that attack **t** arguments.

Example 3.15 (Odd Cycle). Consider an attack cycle between three arguments, the AF $F = (\{a, b, c\}, \{(a, b), (b, c), (c, a)\})$. Its asymmetric conflict-free pairs are given by

$$\begin{array}{llll} (\emptyset, \{a, b, c\}) & (\{a\}, \{a, b, c\}) & (\{b\}, \{a, b, c\}) & (\{c\}, \{a, b, c\}) \\ (\{a\}, \{c, a\}) & (\{b\}, \{a, b\}) & (\{c\}, \{b, c\}) & \end{array}$$

Notice that the admissible pair $(\emptyset, \{a, b, c\})$ is among the asymmetric conflict-free pairs. $\diamond$

It is not hard to show that any admissible pair is asymmetric conflict-free.

Proposition 3.39. *Let $(L, \sqsubseteq)$ be a complete lattice and $\mathcal{O}$ an approximating operator on the bilattice $(L^2, \leq_i)$. Any consistent pair $(x, y) \in L^2$ that is admissible for $\mathcal{O}$ is also asymmetric conflict-free for $\mathcal{O}$.*

Proof. The pair (x, y) is admissible if and only if $x \sqsubseteq \mathcal{O}'(x, y)$ and $\mathcal{O}'(y, x) = \mathcal{O}''(x, y) \sqsubseteq y$. Since $\mathcal{O}'(\cdot, y)$ is $\sqsubseteq$-monotone and $\mathcal{O}'(x, \cdot)$ is $\sqsubseteq$-antimonotone, we obtain the following picture:

$$\begin{array}{ccccccccc} x & \sqsubseteq & \mathcal{O}'(x,y) & & \leqslant & \mathcal{O}'(x,x) & \leqslant & & \\ & & & & & & & \mathcal{O}'(y,x) & \sqsubseteq & y \\ & & & & \leqslant & \mathcal{O}'(y,y) & \leqslant & & \end{array}$$

In particular, $x \sqsubseteq \mathcal{O}''(x, y) = \mathcal{O}'(y, x) \sqsubseteq y$ and (x, y) is asymmetric conflict-free. $\square$

Let us again take note of the asymmetry in the definition of asymmetric conflict-free pairs. If admissible pairs ensure that during revision, both lower and upper bounds are improved, why should asymmetric conflict-free pairs be defined such that only the upper bound must improve? Why not improve the lower bound? Another possibility to define asymmetric conflict-free pairs would have been to say a pair (x, y) is asymmetric conflict-free for an operator $\mathcal{O}$ iff $x \sqsubseteq \mathcal{O}'(x, y) \sqsubseteq y$. In AF terms, this alternative notion allows to set arguments to **u** that are attacked by **t** arguments.

Example 3.16 (Continued from Example 3.15). In the odd attack cycle between three arguments, the alternative asymmetric conflict-free pairs (improving the lower bound) are

$$\begin{array}{llll} (\emptyset, \{a, b, c\}) & (\emptyset, \{a, b\}) & (\emptyset, \{a, c\}) & (\emptyset, \{b, c\}) \\ (\{a\}, \{a, b\}) & (\{b\}, \{b, c\}) & (\{c\}, \{c, a\}) & \end{array}$$ $\diamond$

Unfortunately, this possible alternative version of asymmetric conflict-freeness is not a suitable generalisation of the set-based AF notion, since there are argumentation frameworks with asymmetric conflict-free sets that are not the lower bound of any such pair.

Example 3.17. Consider the argumentation framework $F = (\{a, b\}, \{(a, b)\})$ where a attacks b. The set $X = \{b\}$ is conflict-free. Assume to the contrary that there is a $Y \subseteq A$ such that $X \subseteq U_F(Y) \subseteq Y$. It follows that $b \in U_F(Y)$, that is, Y does not attack b. Thus Y does not contain a and $Y \subseteq \{b\} = X$. From $X \subseteq Y$ we conclude $X = Y = \{b\}$. But we find that $U_F(Y) = \{a, b\} \not\subseteq Y$. Contradiction. $\diamond$

We want to stress that these two different generalisations of conflict-free sets are not an artefact of using approximating operators. Rather, they occur in the step from two-valued to three-valued semantics. As opposed to the alternative, the version requiring improvement of the upper bound generalises conflict-free sets.

Proposition 3.40. *Let $F = (A, R)$ be an AF and $X \subseteq A$. X is conflict-free iff $(X, U_F(X))$ is a asymmetric conflict-free pair.*

Proof.

$$\begin{aligned}
&X \text{ is conflict-free}\\
&\text{iff } X \subseteq U_F(X)\\
&\text{iff } (X, U_F(X)) \text{ is consistent and } X \subseteq \mathcal{F}_F''(X, Y) \subseteq U_F(X)\\
&\text{iff } (X, U_F(X)) \text{ is a asymmetric conflict-free pair}
\end{aligned}$$

□

Indeed, this notion of asymmetric conflict-free pairs for AFs coincides with Caminada's definition of conflict-free labellings (Caminada, 2010).

Proposition 3.41. *For any AF $F = (A, R)$ and $X \subseteq Y \subseteq A$, the pair (X, Y) is asymmetric conflict-free iff the labelling $l : A \to \{\mathbf{t}, \mathbf{f}, \mathbf{u}\}$ with $X \mapsto \mathbf{t}$, $Y \setminus X \mapsto \mathbf{u}$, $A \setminus Y \mapsto \mathbf{f}$ is conflict-free in the sense of (Caminada, 2010, Definition 3).*

3.5.4 Conflict-free (Symmetric)

While the generalisation of conflict-free semantics presented above is a faithful generalisation of conflict-free AF labellings, in this work (especially in the next chapter) we will mainly be interested in a simpler and more intuitive *symmetric* version of three-valued conflict-free semantics.

Definition 3.14. Let $(L, \sqsubseteq)$ be a complete lattice and $\mathcal{O}$ an approximating operator on the bilattice $(L^2, \leq_i)$. A consistent pair $(x, y) \in L^2$ is *symmetric conflict-free for* $\mathcal{O}$ iff $x \sqsubseteq \mathcal{O}''(x, y)$ and $\mathcal{O}'(x, y) \sqsubseteq y$. ◇

This version of conflict-freeness is still a generalisation of the set-based AF notion.

Proposition 3.42. *Let $F = (A, R)$ be an AF, D be its associated ADF and $\mathcal{O} \in \{\mathcal{G}_D, \mathcal{U}_D\}$.*

1. *For each conflict-free set $X \subseteq A$, there exists $Y \subseteq A$ such that (X, Y) is a symmetric conflict-free pair of $\mathcal{O}$.*
2. *For each symmetric conflict-free pair (X, Y), its lower bound X is a conflict-free set.*

Proof. We make use of the fact that for any $P, Q \subseteq A$, we have $\mathcal{O}(P, Q) = (U_F(Q), U_F(P))$, which follows from Proposition 3.23.

1. *Let $X \subseteq A$ be conflict-free. Define $Y = U_F(X)$. Since X is conflict-free,*

   $$X \subseteq Y = U_F(X) = \mathcal{O}''(X, Y)$$

   Furthermore U_F is $\subseteq$-antimonotone, whence $X \subseteq U_F(X)$ implies

   $$\mathcal{O}'(X, Y) = U_F(Y) = U_F(U_F(X)) \subseteq U_F(X) = Y$$

2. *Let (X, Y) be a symmetric conflict-free pair. Then $X \subseteq \mathcal{O}''(X, Y) = U_F(X)$, whence X is a conflict-free set.* □

3.5.5 Naive

It is clear that both types of conflict-free pairs are amenable to the usual maximisation criteria that lead from admissible to preferred and semi-stable semantics. We begin with naive semantics, which for AFs are just $\subseteq$-maximal conflict-free sets.

Definition 3.15. Let $(L, \sqsubseteq)$ be a complete lattice and $\mathcal{O}$ an approximating operator on its corresponding bilattice $(L^2, \leq_i)$. A consistent pair $(x, y) \in L^2$ is an *M-conflict-free pair for* $\mathcal{O}$ iff it is $\leq_i$-maximal among the (symmetric or asymmetric) conflict-free pairs for $\mathcal{O}$. $\diamond$

We keep our uniform naming conventions for recording maximisation criteria. Furthermore, (a)symmetry carries over to naive semantics from the used version of conflict-free semantics. It is straightforward that this definition generalises naive semantics.

Proposition 3.43. *Let $F = (A, R)$ be an AF. A set $X \subseteq A$ is a naive extension of F iff the pair $(X, U_F(X))$ is an M-conflict-free pair of F.*

Proof. First note that for any $X \subseteq A$, the set $U_F(X)$ is fixed. Furthermore U_F is a $\subseteq$-antimonotone operator and thus $X \subseteq Y$ iff $(X, U_F(X)) \leq_i (Y, U_F(Y))$. Consequently for maximisation there is no difference whether we $\subseteq$-maximise the set X or we $\leq_i$-maximise the pair $(X, U_F(X))$. It follows that

X is a naive extension for F
iff X is conflict-free and X is $\subseteq$-maximal
iff $X \subseteq U_F(X)$ and X is $\subseteq$-maximal
iff $X \subseteq U_F(X)$ and $U_F(U_F(X)) \subseteq U_F(X)$ and X is $\subseteq$-maximal
iff $X \subseteq U_F(X) = \mathcal{F}_F''(X, U_F(X))$ and $U_F(U_F(X)) \subseteq U_F(X)$ and $(X, U_F(X))$ is $\leq_i$-maximal
iff $(X, U_F(X))$ is an M-conflict-free pair of $\mathcal{F}_F$ $\square$

3.5.6 Stage

Verheij (1996) defined the notion argumentation stage for an argumentation framework, in turn based on three-valued status assignments. Such an assignment represents not only the arguments that are accepted (as extensions do), but also those which are not accepted. (Hence each assignment gives rise to a unique extension, but not necessarily vice versa.) An *argumentation stage* is a three-valued status assignment to arguments with the restriction that the arguments that are not accepted must be exactly the ones that have an attacker that is accepted. It follows that the set of arguments which are accepted in an argumentation stage (its associated extension) is conflict-free. It is easy to find out how (argumentation) stages can be captured in our setting:

Proposition 3.44. *Let $F = (A, R)$ be an argumentation framework. A consistent pair (X, Y) is an argumentation stage iff $Y = U_F(X)$.*

Proof. (X, Y) is a stage iff $A \setminus Y = R_F(X)$ iff $Y = U_F(X)$. $\square$

While an argumentation stage is always a (symmetric and asymmetric) conflict-free pair, and any conflict-free set gives rise to an argumentation stage (and thus to a conflict-free pair), there are asymmetric conflict-free pairs that are not argumentation stages. (Consider Example 3.17 and the pair $(\{a\}, \{a, b\})$.) Still, we can apply the range maximisation criterion to both types of conflict-free pairs in order to generalise stage extension semantics.

Definition 3.16. Let $(L, \sqsubseteq)$ be a complete lattice and $\mathcal{O}$ an approximating operator on its corresponding bilattice $(L^2, \leq_i)$. A consistent pair $(x, y) \in L^2$ is an *L-conflict-free pair for* $\mathcal{O}$ iff $y \sqcap x^{-1}$ is $\sqsubseteq$-minimal among the conflict-free pairs for $\mathcal{O}$. ◇

For AFs, asymmetric L-conflict-free pairs coincide with stage extensions.

Proposition 3.45. *Let $F = (A, R)$ be an AF. A set $X \subseteq A$ is a stage extension of F iff the pair $(X, U_F(X))$ is an L-stage pair of $\mathcal{F}_F$.*

Proof.

X *is a stage extension for* F
iff X *is conflict-free and* $X \cup R_F(X)$ *is* $\subseteq$*-maximal*
iff X *is conflict-free and* $X \cup \overline{U_F(X)}$ *is* $\subseteq$*-maximal*
iff $X \subseteq U_F(X)$ *and* $\overline{X} \cap U_F(X)$ *is* $\subseteq$*-minimal*
iff $X \subseteq U_F(X) = \mathcal{F}_F''(X, U_F(X))$ *and* $\overline{X} \cap U_F(X)$ *is* $\subseteq$*-minimal*
iff $(X, U_F(X))$ *is an L-stage pair of* $\mathcal{F}_F$ □

3.6 Existence Results for General Operators

We next present two general theorems that guarantee the existence of certain pairs for approximating operators on CPOs. By CPOs here we do in fact refer to arbitrary CPOs $(L^c, \leq_i)$ containing consistent pairs of elements of a complete lattice $(L, \sqsubseteq)$. Both results make use of the axiom of choice – the second one directly, and the first one in the form of Zorn's lemma. The first result says that for each admissible pair there is a preferred pair containing at least as much information. This significantly generalises a result by Dung (1995, Theorem 11) to general operators.

Theorem 3.46. *Let $(L, \sqsubseteq)$ be a complete lattice and $\mathcal{O}$ an approximating operator on the CPO $(L^c, \leq_i)$. For each admissible pair $\bar{a} \in L^c$, there exists a preferred pair $\bar{p} \in L^c$ with $\bar{a} \leq_i \bar{p}$.*

Proof. Let $\bar{a} \in L^c$ with $\bar{a} \leq_i \mathcal{O}(\bar{a})$. Define the set of all $\mathcal{O}$-admissible pairs that contain at least as much information as $\bar{a}$,

$$C = \{\bar{c} \mid \bar{a} \leq_i \bar{c} \text{ and } \bar{c} \leq_i \mathcal{O}(\bar{c})\}$$

We show that $(C, \leq_i)$ is a CPO. Clearly $\bar{a} \in C$ is the least element of the poset $(C, \leq_i)$. Now let $D \subseteq C$ be directed and $\bar{e} = \bigsqcup_{L^c} D$ be its least upper bound in L^c. We show $\bar{e} \in C$, that is, $\bar{a} \leq_i \bar{e}$ and $\bar{e} \leq_i \mathcal{O}(\bar{e})$. Since D is directed, it is non-empty, so there is some $\bar{z} \in D$, whence $\bar{a} \leq_i \bar{z} \leq_i \bar{e}$. Now for each $\bar{z} \in D$, we have $\bar{z} \leq_i \bar{e}$ since $\bar{e}$ is an upper bound of D. Since $\mathcal{O}$ is $\leq_i$-monotone, we have $\mathcal{O}(\bar{z}) \leq_i \mathcal{O}(\bar{e})$. Since $\bar{z} \in D \subseteq C$, by definition $\bar{z} \leq_i \mathcal{O}(\bar{z})$. In combination, $\bar{z} \leq_i \mathcal{O}(\bar{z}) \leq_i \mathcal{O}(\bar{e})$. Thus $\mathcal{O}(\bar{e})$ is an upper bound of D. Since $\bar{e}$ is the least upper bound of D, we have $\bar{e} \leq_i \mathcal{O}(\bar{e})$.

Thus $(C, \leq_i)$ is a CPO and therefore each ascending chain has an upper bound in C. By Zorn's lemma, C has a $\leq_i$-maximal element $\bar{p} \in C$, which by $\bar{a} \leq_i \bar{p}$ is the desired preferred pair. □

For semantics based on symmetric conflict-freeness, an existence result similar to the above Theorem 3.46 holds. The proof follows the proof of Theorem 1 by Bourbaki (1949/50) (see also Theorem 8.23 of Davey and Priestley, 2002, in particular for the concept of "roofs"), and sufficiently complicated. The major part of the proof is concerned with showing that there is a chain of symmetric conflict-free elements that starts with the given symmetric conflict-free element, and that this chain is itself a CPO. Again, the result is not restricted to subset-CPOs.

Theorem 3.47. *Let $(L, \sqsubseteq)$ be a complete lattice and $\mathcal{O}$ an approximating operator on the CPO $(L^c, \leq_i)$. For each symmetric conflict-free pair $\bar{c} \in L^c$, there exists a (symmetric) naive pair $\bar{n} \in L^c$ with $\bar{c} \leq_i \bar{n}$.*

Proof. Let $\bar{c} \in L^c$ be conflict-free. Define the set

$$D = \{\bar{a} \in L^c \mid \bar{c} \leq_i \bar{a}\}$$

Clearly $(D, \leq_i)$ is a CPO with least element $\bar{c}$. (Its least upper bound is given by $\sqcup_D = \sqcup_{L^c}$.) For any conflict-free pair $\bar{a} \in D$ that is not naive, by definition there exists a conflict-free pair $\bar{a}' \in D$ such that $\bar{a} <_i \bar{a}'$. Thus by the axiom of choice, there exists a function $f : D \to D$ with

$$\bar{a} \mapsto \begin{cases} \bar{a}' & \text{if } \bar{a} \text{ is conflict-free, but not naive} \\ \bar{a} & \text{otherwise} \end{cases}$$

Clearly f is increasing, that is, for all $\bar{a} \in D$ we have $\bar{a} \leq_i f(\bar{a})$. Furthermore, $f(\bar{a})$ is conflict-free iff $\bar{a}$ is conflict-free. Thus a conflict-free pair $\bar{a}$ is a fixpoint of f iff $\bar{a}$ is naive. We proceed to show that such a fixpoint exists.

We look at the smallest f-closed sub-CPO of $(D, \leq_i)$, that is, the $\subseteq$-least set $F \subseteq D$ such that $f(F) \subseteq F$ and $(F, \leq_i)$ is a CPO. Clearly its least element is $\bot_F = \bar{c}$, the least element of D.

We call an element $\bar{u} \in F$ a roof *iff for all $\bar{v} \in F$ with $\bar{v} <_i \bar{u}$ we have $f(\bar{v}) \leq_i \bar{u}$. For each pair $\bar{u} \in F$, we show that if $\bar{u}$ is a roof, then the set*

$$Z_{\bar{u}} = \{\bar{v} \in F \mid \bar{v} \leq_i \bar{u} \text{ or } f(\bar{u}) \leq_i \bar{v}\}$$

is an f-closed sub-CPO of $(F, \leq_i)$. So let $\bar{u} \in F$ be a roof and consider $Z_{\bar{u}}$. We have to show that $f(Z_{\bar{u}}) \subseteq Z_{\bar{u}}$ and $(Z_{\bar{u}}, \leq_i)$ is a CPO.

$f(Z_{\bar{u}}) \subseteq Z_{\bar{u}}$: Let $\bar{v} \in Z_{\bar{u}}$. Then $\bar{v} \leq_i \bar{u}$ or $f(\bar{u}) \leq_i \bar{v}$. We have to show $f(\bar{v}) \in Z_{\bar{u}}$, that is, $f(\bar{v}) \leq_i \bar{u}$ or $f(\bar{u}) \leq_i f(\bar{v})$. If $f(\bar{u}) \leq_i \bar{v}$, then since f is increasing we get $f(\bar{u}) \leq_i \bar{v} \leq_i f(\bar{v})$. If $\bar{v} <_i \bar{u}$, then since $\bar{u}$ is a roof we get $f(\bar{v}) \leq_i \bar{u}$. If $\bar{v} = \bar{u}$ then $f(\bar{u}) \leq_i f(\bar{v})$ is clear.

$(Z_{\bar{u}}, \leq_i)$ is a CPO: $\bot_F \in Z_{\bar{u}}$ is the least element of the poset $(Z_{\bar{u}}, \leq_i)$. Let $E \subseteq Z_{\bar{u}}$ be directed and $\bar{e} = \bigsqcup_F E$ be its least upper bound in $(F, \leq_i)$. We have to show $\bar{e} \in Z_{\bar{u}}$, that is, $\bar{e} \leq_i \bar{u}$ or $f(\bar{u}) \leq_i \bar{e}$. By assumption,

$$Z_{\bar{u}} = Z^l_{\bar{u}} \cup Z^r_{\bar{u}} \text{ with } Z^l_{\bar{u}} = \{\bar{v} \in F \mid \bar{v} \leq_i \bar{u}\} \text{ and } Z^r_{\bar{u}} = \{\bar{v} \in F \mid f(\bar{u}) \leq_i \bar{v}\}$$

Define $E^l = E \cap Z^l_{\bar{u}}$ and $E^r = E \cap Z^r_{\bar{u}}$. Clearly $\bar{u}$ is an upper bound of E^l and $f(\bar{u})$ is a lower bound of E^r; moreover $\bar{e}$ is an upper bound of E^r. Thus if $E^r \neq \emptyset$ then $f(\bar{u}) \leq_i \bar{e}$ and we are done. Otherwise $E^r = \emptyset$, then $E = E^l$ and $\bar{u}$ is an upper bound of E. Since $\bar{e}$ is the least upper bound of E, we get $\bar{e} \leq_i \bar{u}$.

Thus if $\bar{u} \in F$ is a roof then $(Z_{\bar{u}}, \leq_i)$ with $Z_{\bar{u}} \subseteq F$ is an f-closed sub-CPO of $(D, \leq_i)$. Since $(F, \leq_i)$ is the least f-closed sub-CPO of $(D, \leq_i)$, we get $F \subseteq Z_{\bar{u}}$ and thus $Z_{\bar{u}} = F$ for each roof $\bar{u} \in F$. Now we show that each pair $\bar{u} \in F$ is a roof. Define the set $U = \{\bar{u} \in F \mid \bar{u} \text{ is a roof}\}$. We show that $(U, \leq_i)$ is an f-closed sub-CPO of $(F, \leq_i)$.

$f(U) \subseteq U$: Let $\bar{u} \in U$. Then for all $\bar{v} \in F$ with $\bar{v} <_i \bar{u}$ we have $f(\bar{v}) \leq_i \bar{u}$. We have to show $f(\bar{u}) \in U$, that is, for all $\bar{v} \in F$ with $\bar{v} <_i f(\bar{u})$ we have $f(\bar{v}) \leq_i f(\bar{u})$.

Let $\bar{v} \in F$ with $\bar{v} <_i f(\bar{u})$. Since $\bar{v} \in F = Z_{\bar{u}}$, we find that $\bar{v} \leq_i \bar{u}$ or $f(\bar{u}) \leq_i \bar{v}$. Note that $f(\bar{u}) \leq_i \bar{v}$ is impossible by presumption. If $\bar{v} <_i \bar{u}$ then we have $f(\bar{v}) \leq_i \bar{u} \leq_i f(\bar{u})$ by presumption. If $\bar{v} = \bar{u}$ then $f(\bar{v}) \leq_i f(\bar{u})$ is clear.

$(U, \leq_i)$ is a CPO: *$\bot_F$ is trivially a roof, whence $\bot_F \in U$. Now let $W \subseteq U$ be directed and let $\bar{w} = \bigsqcup_F W$ be the least upper bound of W in F. We show $\bar{w} \in U$, that is, for all $\bar{v} \in F$ with $\bar{v} <_i \bar{w}$ we have $f(\bar{v}) \leq_i \bar{w}$.*

Let $\bar{v} \in F$ with $\bar{v} <_i \bar{w}$. If for all $\bar{z} \in W$ we had $\bar{z} \leq_i \bar{v}$, then $\bar{v}$ would be an upper bound of W, whence $\bar{w} \leq_i \bar{v}$ contrary to assumption. Thus there is a $\bar{z} \in W$ with $\bar{z} \not\leq_i \bar{v}$. Now $\bar{z} \in W \subseteq U$ is a roof, and we have $\bar{v} \in F = Z_{\bar{z}}$, that is, $\bar{v} \leq_i \bar{z}$ or $f(\bar{z}) \leq_i \bar{v}$. Due to $\bar{z} \leq_i f(\bar{z})$ and $\bar{z} \not\leq_i \bar{v}$ we get $\bar{v} \leq_i \bar{z}$; additionally, $\bar{z} \leq_i \bar{w}$ since $\bar{w}$ is an upper bound of W. Now if $\bar{v} = \bar{z}$ then $\bar{v}$ is a roof and $\bar{w} \leq_i \bar{v}$ or $f(\bar{v}) \leq_i \bar{w}$, where the first is impossible by presumption. Finally, if $\bar{v} <_i \bar{z}$ then $\bar{z}$ being a roof implies that $f(\bar{v}) \leq_i \bar{z} \leq_i \bar{w}$.

Thus $(U, \leq_i)$ with $U \subseteq F$ is an f-closed sub-CPO of $(D, \leq_i)$. Since $(F, \leq_i)$ is the least f-closed sub-CPO of $(D, \leq_i)$, we have $F \subseteq U$, that is, $F = U$.

Now we show that F is a chain, that is, for all $\bar{u}, \bar{v} \in F$ we find $\bar{u} \leq_i \bar{v}$ or $\bar{v} \leq_i \bar{u}$: since $\bar{u}$ is a roof, $\bar{v} \in F = Z_{\bar{u}}$ whence $\bar{v} \leq_i \bar{u}$ or $\bar{u} \leq_i f(\bar{u}) \leq_i \bar{v}$. Now F is a CPO and a chain, it therefore has a least upper bound in F, that is, a greatest element $\top_F = \bigsqcup_F F$. Since f is increasing, we have $\top_F \leq_i f(\top_F)$; since F is f-closed, $f(\top_F) \in F$; since $\top_F$ is the greatest element of F, we find $f(\top_F) \leq_i \top_F$. Thus $\top_F$ is a fixpoint of f. It remains to show that $\top_F$ is conflict-free. In fact, all elements of F are conflict-free: assume there were a $\bar{v} \in F$ that was not conflict-free, then $f^{-1}(\bar{v}) = \{\bar{v}\}$ by definition of f and the pair $(F \setminus \{\bar{v}\}, \leq_i)$ would be an f-closed proper sub-CPO of F, contradiction. Consequently, $\bar{n} = \top_F$ with $\bar{c} = \bot_F \leq_i \top_F = \bar{n}$ is our desired naive pair. □

From the last part of the proof it might seem that the desired naive pair is uniquely determined. This is however not the case – the application of the axiom of choice in the beginning gives us an arbitrary chain of conflict-free pairs, there might be many more in $(L^c, \leq_i)$.

We finally prove a useful technical result that gives some insight into the structure of sets of symmetric conflict-free interpretations, namely, that such sets are downward-closed with respect to the CPO ordering. Notably, again, this result holds for arbitrary approximating operators.

Lemma 3.48. *Let $(L, \sqsubseteq)$ be a complete lattice and $\mathcal{O}$ an approximating operator on the CPO $(L^c, \leq_i)$. If $(x, y) \in L^c$ is symmetric conflict-free for $\mathcal{O}$, then so is any $(u, v) \leq_i (x, y)$.*

Proof. Let $(x, y) \in L^c$ be symmetric conflict-free for $\mathcal{O}$ and $(u, v) \leq_i (x, y)$. First observe that this means $x \sqsubseteq \mathcal{O}''(x, y)$, $\mathcal{O}'(x, y) \sqsubseteq y$ and $u \sqsubseteq x \sqsubseteq y \sqsubseteq v$. Now since $\mathcal{O}$ is approximating, it is in particular $\leq_i$-monotone and thus $\mathcal{O}(u, v) \leq_i \mathcal{O}(x, y)$, that is,

$$\mathcal{O}'(u, v) \sqsubseteq \mathcal{O}'(x, y) \quad \text{and} \quad \mathcal{O}''(x, y) \sqsubseteq \mathcal{O}''(u, v)$$

Combining all of the above, it follows that

$$u \sqsubseteq x \sqsubseteq \mathcal{O}''(x, y) \sqsubseteq \mathcal{O}''(u, v)$$
$$\mathcal{O}'(u, v) \sqsubseteq \mathcal{O}'(x, y) \sqsubseteq y \sqsubseteq v$$

whence (u, v) is symmetric conflict-free for $\mathcal{O}$. □

3.7 Overview of Results

In this chapter, we embedded abstract dialectical frameworks into Denecker et al.'s lattice-theoretical formalism for the abstract study of logical languages. This provides useful insights

into the relationship of abstract argumentation frameworks and abstract dialectical frameworks with other logic-based knowledge representation formalisms.

In this last section, we will provide a concise overview over the results of our investigation. First, for reference and as a completion of the table in Definition 2.2, we review the definitions of operator-based semantics in Table 3.1.

symmetric conflict-free pair (x,y)	$x \sqsubseteq \mathcal{O}''(x,y)$ and $\mathcal{O}'(x,y) \sqsubseteq y$
M-symmetric conflict-free pair (x,y)	$x \sqsubseteq \mathcal{O}''(x,y)$ and $\mathcal{O}'(x,y) \sqsubseteq y$ and (x,y) is $\leq_i$-maximal
L-symmetric conflict-free pair (x,y)	$x \sqsubseteq \mathcal{O}''(x,y)$ and $\mathcal{O}'(x,y) \sqsubseteq y$ and $y \sqcap x^{-1}$ is $\sqsubseteq$-minimal
asymmetric conflict-free pair (x,y)	$x \sqsubseteq \mathcal{O}''(x,y) \sqsubseteq y$
M-asymmetric conflict-free pair (x,y)	$x \sqsubseteq \mathcal{O}''(x,y) \sqsubseteq y$ and (x,y) is $\leq_i$-maximal
L-asymmetric conflict-free pair (x,y)	$x \sqsubseteq \mathcal{O}''(x,y) \sqsubseteq y$ and $y \sqcap x^{-1}$ is $\sqsubseteq$-minimal
admissible/reliable pair (x,y)	$(x,y) \leq_i \mathcal{O}(x,y)$
Kripke-Kleene semantics	$lfp(\mathcal{O})$
three-valued supported model (x,y)	$\mathcal{O}(x,y) = (x,y)$
M-supported model (x,y)	$\mathcal{O}(x,y) = (x,y)$ and (x,y) is $\leq_i$-maximal
L-supported model (x,y)	$\mathcal{O}(x,y) = (x,y)$ and $y \sqcap x^{-1}$ is $\sqsubseteq$-minimal
two-valued supported model (x,x)	$\mathcal{O}(x,x) = (x,x)$
well-founded semantics	$lfp(\mathcal{SO})$
three-valued stable model (x,y)	$\mathcal{SO}(x,y) = (x,y)$
M-stable model (x,y)	$\mathcal{SO}(x,y) = (x,y)$ and (x,y) is $\leq_i$-maximal
L-stable model (x,y)	$\mathcal{SO}(x,y) = (x,y)$ and $y \sqcap x^{-1}$ is $\sqsubseteq$-minimal
two-valued stable model (x,x)	$\mathcal{SO}(x,x) = (x,x)$

Table 3.1: Operator-based semantical notions. All of them are defined for $x, y \in L$ with $x \sqsubseteq y$ for complete lattices $(L, \sqsubseteq)$ and approximating operators $\mathcal{O}$ on their corresponding bilattice, in some cases (L-supported, L-stable) with additional restrictions on join, meet and involution operations on the lattice.

Figure 3.1 then depicts the relationship between the different semantical notions explored in this chapter. If a semantics σ is seen as a function assigning to a knowledge base κ a set of models, then a partial order on semantics is given by $\sigma_1 \leq \sigma_2$ iff $\sigma_1(\kappa) \subseteq \sigma_2(\kappa)$ for all κ. In the figure, an arrow from σ_1 to σ_2 expresses $\sigma_1 \leq \sigma_2$ – in words, all σ_1-models are also σ_2-models.

Next, Table 3.2 shows the correspondences between different argumentation semantics and operator-based semantics. The operator-based semantics lead to new semantics for default logic and autoepistemic logics via their respective consequence operators (Denecker et al., 2003).

3.8 Concluding Remarks

The several new correspondence results for AFs and logic programs we proved extended results of Wu et al. (2009), who showed correspondence of complete extensions and three-valued stable models. While the results of Wu et al. (2009) use the translation of Gabbay and d'Avila Garcez (2009), they do not motivate the use of this – we call it standard – translation nor provide a comparison to the much older Dung translation. In this work we showed that using the standard translation is justified; what is more, we even proved that the standard translation and Dung's translation produce equivalent programs.

Concerning translations from AFs into LPs, related work has also been done by a number of authors, albeit with different goals: Both Wakaki and Nitta (2008) and Egly, Gaggl, and Woltran (2010) want to efficiently implement different argumentation semantics using the stable model semantics for logic programming. Furthermore they employ meta-programming and answer set programming with variables to allow for modular translations. Toni and Sergot

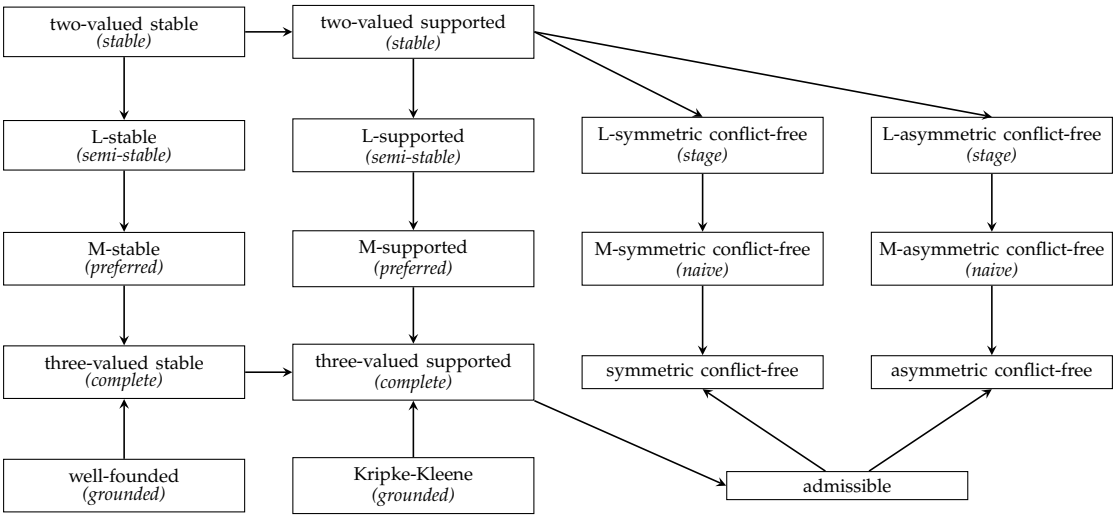


Figure 3.1: Inclusion relations between operator-based semantics. Nodes depict semantical notions for elements of a bilattice, where the names in parentheses are argumentation versions of these notions. Directed edges indicate subset relationships between the sets of all bilattice elements which satisfy the respective semantical notion. For example, the arrow from two-valued stable to two-valued supported in the upper left corner means that all two-valued stable models are also two-valued supported models.

(2011) survey these and other uses of (forms of) answer set programming to implement abstract argumentation semantics. Nieves, Osorio, and Zepeda (2011) define new argumentation semantics that possess certain desired properties. They do this by first providing a general recursive schema for obtaining new logic programming semantics and then defining the argumentation semantics via the AFs' translated logic programs. The translation from AFs into LPs they use is very similar to the one of Dung (1995).

Besnard and Doutre (2004) redefined argumentation semantics in terms of fixpoints, but they do not look at grounded or semi-stable semantics and do not use their insights to embed argumentation frameworks into the larger picture. Grossi (2012) investigated fixpoint-based definitions of argumentation semantics to study the connection between argumentation and dynamic epistemic logic.

In general, we are not aware of any works that address the relationship of abstract dialectical frameworks with other nonmonotonic knowledge representation formalisms, attempt a principled reconstruction of ADF semantics or generalise argumentation semantics to an abstract operator-based setting.

As we observed in Example 3.10, it is not entirely clear how to define the union of two ADFs that share statements. Although for specific representations of acceptance conditions such a union should be straightforward to define, we want to devote some future work into abstracting from specific representations and develop a general method for combining ADFs.

Corollary 3.13 has shown that Brewka and Woltran (2010) defined not only the notion of an ADF model, but also the ultimate three-valued approximation of this notion. Furthermore, Denecker et al. (2004) study several other ultimate semantics. It is an important aspect of future work to investigate these ultimate semantics in detail and to compare them with the ones investigated here and by Brewka et al. (2013).

We remarked on several occasions throughout Chapter 3 that we defined new semantics for

Operator	AF	ADF
symmetric conflict-free pair	conflict-free set	**symmetric conflict-free pair**
M-symmetric conflict-free pair	naive extension	**M-symmetric conflict-free pair**
L-symmetric conflict-free pair	stage extension	**L-symmetric conflict-free pair**
asymmetric conflict-free pair	conflict-free set/labelling	**asymmetric conflict-free pair**
M-asymmetric conflict-free pair	naive extension	**M-asymmetric conflict-free pair**
L-asymmetric conflict-free pair	stage extension	**L-asymmetric conflict-free pair**
reliable pair	admissible set	**admissible pair**
Kripke-Kleene semantics	grounded extension	**Kripke-Kleene semantics**
three-valued supported model	complete extension	**three-valued supported model**
M-supported model	preferred extension	**M-supported model**
L-supported model	semi-stable extension	**L-supported model**
two-valued supported model	stable extension	(two-valued supported) model
well-founded semantics	grounded extension	**well-founded semantics**
three-valued stable model	complete extension	**three-valued stable model**
M-stable model	preferred extension	**M-stable model**
L-stable model	semi-stable extension	**L-stable model**
two-valued stable model	stable extension	**two-valued stable model**

*Table 3.2: Overview over semantics for approximating operators, argumentation frameworks and abstract dialectical frameworks. Semantics newly defined in this thesis are written in **bold font**. Most extension semantics for AFs have at least two generalisations, a supported and a stable one. While most argumentation semantics already had a corresponding operator semantics, we found that (symmetric/asymmetric) conflict-free pairs and maximisation versions of these lead to new semantical notions for approximating operators. The operator-based versions of argumentation semantics then directly lead to the ADF generalisations of these semantics, most of which are newly defined in this work. M/L-stable/supported models for operators are straightforwardly generalised notions from logic programming. Operator-based semantics then immediately lead to semantics for default logic and autoepistemic logic (not included in this presentation).*

default and autoepistemic logics (admissible, preferred, semi-stable, stage). These semantics could be studied in greater detail, especially their strengths and weaknesses in comparison to the standard semantics of these two nonmonotonic KR formalisms. Additionally, in the same way we defined several semantics for normal logic programs (conflict-free, admissible, naive, stage). In order to determine whether these semantics are new, it might be a good starting point to compare them to the semantics discussed by Eiter, Fink, and Moura (2010).

Chapter 4

Computational Complexity

For future successful applications of ADFs, it is indispensable to figure out which semantics is suitable for which type of application. There is a great number of semantics for AFs already, and many of them have been generalised to ADFs. Thus it might not be clear to potential ADF users which semantics are adequate for a particular application domain. In general, abstract argumentation is still a young field and has not yet found conclusive reasons to definitely prefer some semantics over others. In this regard, knowing the computational complexity of semantics can be a valuable guide. However, previously existing complexity results for ADFs were scattered over different papers, missed several semantics and some of them presented upper bounds only. In this chapter, we provide a comprehensive complexity analysis for ADFs. In line with the literature, we represent acceptance conditions by propositional formulas as they provide a compact and elegant way to represent Boolean functions.

Technically, we base our complexity analysis on the approximation fixpoint theory (AFT) by Denecker, Marek and Truszczyński (Denecker et al., 2000, 2003, 2004). AFT embodies the intuitions of decades of KR research; we believe that this is very valuable also for relatively recent languages (such as ADFs), because we get the enormously influential formalisations of intuitions of Reiter and others for free. (As a liberal variation on Newton, we could say that approximation fixpoint theory allows us to take the elevator up to the shoulders of giants instead of walking up the stairs.) In fact, approximation fixpoint theory can be and partially has already been used to define some of the semantics of ADFs, namely in Chapter 3 of this thesis. There, we generalised various AF and logic programming semantics to ADFs using AFT, which has provided us with two families of semantics, *approximate* and *ultimate* versions of semantics, respectively. Intuitively speaking, both families approximate the original two-valued model semantics of ADFs, where the ultimate family is more *precise* in a formally defined sense. The present chapter employs approximating operators for complexity analysis and thus shows that AFT is also well-suited for studying the computational complexity of formalisms.

Along with providing a comparison of the approximate and ultimate families of semantics, our main results can be summarised as follows. We show that: (1) the computational complexity of ADF decision problems is one level up in the polynomial hierarchy from their AF counterparts (Dunne and Wooldridge, 2009); (2) the ultimate semantics are almost always as complex as the approximate semantics, with the notable exceptions of two-valued stable models, and (symmetric) conflict-free and naive semantics; (3) there is a certain subclass of ADFs, called *bipolar* ADFs (BADFs), which is of the same complexity as AFs (with the single exception of sceptical reasoning for naive semantics). Intuitively, in bipolar ADFs all links between

statements are *supporting* or *attacking*. To formalise these notions, Brewka and Woltran (2010) gave a precise semantical definition of support and attack. In our work, we assume that the link types are specified by the user along with the ADF. We consider this a harmless assumption since the existing applications of ADFs produce bipolar ADFs where the link types are known (Brewka and Gordon, 2010; Strass, 2015c). This attractiveness of bipolar ADFs from a KR point of view is the most significant result of this chapter: it shows that BADFs offer – in addition to AF-like and more general notions of attack – also syntactical notions of support *without any increase in computational cost*.

In BADFs, support for a statement s can be anything among "set support" (all statements in a certain set must be accepted for the support to become active) or "individual support" (at least one statement supporting s must be accepted for the support to become active). In the same vein, BADFs offer "set attack" (all statements in a certain set must be accepted for the attack to become active) and the traditional "individual attack" known from AFs (at least one statement attacking s must be accepted for the attack to become active). Naturally, in BADFs all these different notions of support and attack can be freely combined.

Previously, Brewka et al. (2011) translated BADFs into AFs for two-valued semantics and suggested indirectly that the complexities align.[1] Here we go a direct route, which has more practical relevance since it immediately affects algorithm design. Our work was also inspired by the complexity analysis of assumption-based argumentation by Dimopoulos, Nebel, and Toni (2002) – they derived generic results in a way similar to ours.

Our complexity results aligning AFs and BADFs are especially remarkable with regard to expressiveness in the model-theoretic sense. While it remains elusive what kinds of sets of two-valued interpretations the class of AFs can express exactly (Baumann, Dvořák, Linsbichler, Strass, and Woltran, 2014; Baumann, Dvořák, Linsbichler, Spanring, Strass, and Woltran, 2016), we know that even bipolar ADFs can express strictly more than that (at least all $\subseteq$-antichains), and general (non-bipolar) ADFs can express any set of two-valued interpretations with the two-valued model semantics (Chapter 5). This shows that AFs (under stable extension/labelling semantics) – while being of equal computational complexity – are strictly less expressive than (B)ADFs (under model semantics, one of the ADF counterparts of AF stable semantics).

In this chapter, we will not consider the asymmetric version of conflict-free semantics at all. Complexity results for that family of semantics are indeed substantially different and can be found in the work of Gaggl, Rudolph, and Strass (2015). So in this chapter, whenever we write conflict-free, naive, and stage we implicitly mean symmetric conflict-free, naive, and stage.

One important proof technique of this chapter is to employ ADFs' acceptance conditions' representation via propositional formulas and to partially evaluate them. For a propositional formula φ over vocabulary P and $X \subseteq Y \subseteq P$ we define the *partial valuation of φ by (X, Y)* as

$$\varphi^{(X,Y)} = \varphi[p/\top : p \in X][p/\bot : p \in P \setminus Y]$$

Intuitively, the pair (X, Y) represents a partial interpretation of P where all elements of X are true and all elements of $P \setminus Y$ are false.[2] The partial evaluation of φ with (X, Y) takes the two-valued part of (X, Y) and replaces the evaluated variables by their truth values. Naturally, $\varphi^{(X,Y)}$ is a formula over the vocabulary $Y \setminus X$, that is, only contains variables that have no classical truth value (true or false) in the pair (X, Y). In particular, for any total interpretation (X, X), the partial evaluation $\varphi^{(X,X)}$ is a Boolean expression consisting only of truth constants and connectives and thus has a fixed truth value (either true or false).

[1] Additionally, in contrast to Brewka et al. (2011), we use a revised version of the stable model semantics.

[2] Equivalently, the pair (X, Y) represents a three-valued interpretation where all elements of $Y \setminus X$ are undefined.

We will show that approximate and ultimate ADF operators (and thus all of the operator-based ADF semantics) can be defined in terms of partial evaluations of acceptance formulas. For example, in the new three-valued conflict-free semantics that we introduce, a statement s can only be set to true in an interpretation (X, Y) if the partial evaluation of its acceptance formula with the interpretation – the formula $\varphi_s^{(X,Y)}$ – is satisfiable. Symmetrically, s can only be set to false in (X, Y) if $\varphi_s^{(X,Y)}$ is refutable. For the three-valued admissible semantics, the justification standards are higher. There, setting s to true is only justified if $\varphi_s^{(X,Y)}$ is irrefutable (a tautology), setting s to false is only justified if $\varphi_s^{(X,Y)}$ is unsatisfiable. This logical view of (argumentation) semantics thus provides a novel perspective on notions of acceptability.

This chapter proceeds as follows. In the next section, we define the relevant decision problems, use examples to illustrate how operators revise ADF interpretations and show generic upper complexity bounds along with some other useful preparatory technical results. In the main section on complexity results for general ADFs, we back up the upper bounds with matching lower bounds; the section afterwards does the same for bipolar ADFs. We end this chapter with a brief discussion of related and future work.

4.1 Preparatory Considerations

This section sets the stage and provides several technical preparations that will simplify our complexity analysis that follows afterwards. We first introduce some notation to make formally precise what decision problems we will analyse (Section 4.1.1). Next, in Section 4.1.2 we study the relationship between the approximate and ultimate operator, where it will turn out that the operators are quite similar, yet subtly different.

Since several of our hardness results use similar reduction techniques, we introduce some of them in Section 4.1.3 and prove properties that we will later use in hardness proofs. In Section 4.1.4 we analyse the complexity of computing the two operators we consider in this chapter. Since the semantics that we study are defined within the framework of approximation fixpoint theory, knowing the complexity of operator computation is a valuable guide for investigating the operator-based semantics. Finally, in Section 4.1.5 we give generic results on upper bounds for operator-based semantics that only make use of upper bounds for the respective operators.

4.1.1 Notation and Decision Problems

For a set S, we denote by

- $(A^c, \leq_i)$ the consistent CPO of S-subset pairs,
- $\mathcal{O}$ an approximating operator on $(A^c, \leq_i)$.

In the following we tacitly assume that from a given approximation operator $\mathcal{O}$ one can infer the context CPO and the underlying set S, unless noted otherwise.

Let $\mathcal{A}$ be the set of all approximation operators, such that each is defined on some consistent CPO of S-subset pairs for some set S. We define decision problems with two parameters. The first is a set of approximation operators $\mathcal{I} \subseteq \mathcal{A}$. In addition to $\mathcal{A}$ we are interested in this chapter in the following sets of operators.

- $\mathcal{G} = \{\mathcal{G}_D \mid D \text{ is an ADF}\}$,

- $\mathcal{U} = \{\mathcal{U}_D \mid D \text{ is an ADF}\}$

That is, the sets contain approximate, respectively ultimate operators for each possible ADF. When restricted to *bipolar* ADFs we denote the corresponding sets with

- $\mathcal{BG} = \{\mathcal{G}_D \mid D \text{ is a BADF}\}$,
- $\mathcal{BU} = \{\mathcal{U}_D \mid D \text{ is a BADF}\}$.

Clearly we have $\mathcal{G}, \mathcal{U} \subseteq \mathcal{A}$ and thus also $\mathcal{BG}, \mathcal{BU} \subseteq \mathcal{A}$. The semantics is the second parameter of our decision problems. Let $\sigma \in \{scf, nai, adm, com, grd, pre, mod, stm\}$ be a semantics among symmetric conflict-free (Definition 3.14), naive, admissible, complete, grounded, preferred, two-valued supported and two-valued stable semantics, respectively.

We first consider the *verification* problem, which asks if for a given operator a given pair is a σ-pair, respectively a σ-model.

Problem: $\mathsf{Ver}_\sigma^{\mathcal{I}}$
Instance: An approximation operator $\mathcal{O} \in \mathcal{I}$ and a pair $(X, Y) \in A^c$.
Question: Is (X, Y) a σ-model/pair of $\mathcal{O}$?

For instance $\mathsf{Ver}_{adm}^{\mathcal{G}}$ asks whether for a given approximate operator $\mathcal{G}_D$ and $(X, Y) \in A^c$, does it hold that $(X, Y) \leq_i \mathcal{G}_D(X, Y)$? The next decision problem asks whether there *exists a non-trivial* σ-pair/model, that is, one that is different from $(\emptyset, S)$.

Problem: $\mathsf{Exists}_\sigma^{\mathcal{I}}$
Instance: An approximation operator $\mathcal{O} \in \mathcal{I}$.
Question: Does there exist a σ-model/pair (X, Y) of $\mathcal{O}$ such that $(X, Y) \neq (\emptyset, S)$?

The remaining two decision problems define query-based reasoning. The *credulous* acceptance problem asks whether an element $s \in S$ is in X of at least one σ-pair/model (X, Y) of a given operator, while *sceptical* acceptance asks if this is the case for all σ-pairs/models.

Problem: $\mathsf{Cred}_\sigma^{\mathcal{I}}$
Instance: An approximation operator $\mathcal{O} \in \mathcal{I}$ and $s \in S$.
Question: Does there exist a σ-model/pair (X, Y) of $\mathcal{O}$ such that $s \in X$?

Problem: $\mathsf{Scep}_\sigma^{\mathcal{I}}$
Instance: An approximation operator $\mathcal{O} \in \mathcal{I}$ and $s \in S$.
Question: Does it hold that for all σ-models/pairs (X, Y) of $\mathcal{O}$ we have $s \in X$?

We now introduce auxiliary decision problems, which aid us in showing the computational complexity of revising the lower and upper bounds for a given approximation operator and pair. The first one asks whether an element is in the revised lower bound (respectively upper bound) for a given pair.

Problem: $\mathsf{Elem}^{\mathcal{I}'}$ (resp. $\mathsf{Elem}^{\mathcal{I}''}$)
Instance: An approximation operator $\mathcal{O} \in \mathcal{I}$, a pair $(X, Y) \in A^c$ and $s \in S$.
Question: Does it hold that $s \in \mathcal{O}'(X, Y)$ (resp. $s \in \mathcal{O}''(X, Y)$)?

Let $\circ \in \{\subseteq, \supseteq\}$. The next decision problem considers all combinations of asking whether for a given pair and approximation operator the given set is a subset/superset of the revised lower/upper bound.

Problem: $\mathsf{RevBound}_{\circ}^{\mathcal{I}'}$
Instance: An approximation operator $\mathcal{O} \in \mathcal{I}$, a pair $(X, Y) \in A^c$ and a set $B \subseteq S$.
Question: if $\circ = \subseteq$: Is $B \subseteq \mathcal{O}'(X, Y)$?
if $\circ = \supseteq$: Is $\mathcal{O}'(X, Y) \subseteq B$?

Similarly, $\mathsf{RevBound}_{\circ}^{\mathcal{I}''}$ denotes the variant for the revision of the upper bound ($\mathcal{O}''$). For instance $\mathsf{RevBound}_{\supseteq}^{\mathcal{I}''}$ denotes the problem of checking whether for an approximation operator $\mathcal{O} \in \mathcal{I}$, $B \subseteq S$ and a given pair $(X, Y) \in A^c$ we have $\mathcal{O}''(X, Y) \subseteq B$, that is, if the set is a superset of the revised upper bound (indicated by $\cdot''$).

4.1.2 Relationship Between the Operators

Since $\mathcal{U}_D$ is the ultimate approximation of G_D for an ADF D it is clear that for any $X \subseteq Y \subseteq S$ we have $\mathcal{G}_D(X, Y) \leq_i \mathcal{U}_D(X, Y)$. In other words, the ultimate revision operator produces new bounds that are at least as tight as those of the approximate operator. More explicitly, the ultimate new lower bound always contains the approximate new lower bound: $\mathcal{G}'_D(X, Y) \subseteq \mathcal{U}'_D(X, Y)$; conversely, the ultimate new upper bound is contained in the approximate new upper bound: $\mathcal{U}''_D(X, Y) \subseteq \mathcal{G}''_D(X, Y)$. Somewhat surprisingly, it turns out that the revision operators for the upper bound coincide.

Lemma 4.1. *Let $D = (S, L, C)$ be an ADF and $X \subseteq Y \subseteq S$.*

$$\mathcal{G}''_D(X, Y) = \mathcal{U}''_D(X, Y)$$

Proof. Let $s \in S$. We will use that for all $B, X, P \subseteq S$, we find $(P \setminus B) \cap X = \emptyset$ iff $P \cap X \subseteq B$. Now

$$\begin{aligned} s \in \mathcal{G}''_D(X, Y) \text{ iff } & \exists B : B \subseteq par(s) \cap Y \text{ and } C_s(B) = \mathbf{t} \text{ and } (par(s) \setminus B) \cap X = \emptyset \\ \text{iff } & \exists B : par(s) \cap X \subseteq B \subseteq par(s) \cap Y \text{ and } C_s(B) = \mathbf{t} \\ \text{iff } & \exists Z : X \subseteq Z \subseteq Y \text{ and } C_s(Z \cap par(s)) = \mathbf{t} \\ \text{iff } & s \in \mathcal{U}''_D(X, Y) \end{aligned}$$

□

The operators for computing a new lower bound are demonstrably different, since we can find D and (X, Y) with $\mathcal{U}'_D(X, Y) \not\subseteq \mathcal{G}'_D(X, Y)$, as the following ADF shows.

Example 4.1. Consider the ADF $D = (\{a\}, \{(a, a)\}, \{\varphi_a\})$ with one self-dependent statement a that has acceptance formula $\varphi_a = a \vee \neg a$. In Figure 4.1, we show the relevant CPO and the behaviour of approximate and ultimate operators: we see that $\mathcal{G}_D(\emptyset, \{a\}) <_i \mathcal{U}_D(\emptyset, \{a\})$, which shows that in some cases the ultimate operator is strictly more precise. ◇

So in a sense the approximate operator cannot see beyond the case distinction $a \vee \neg a$. As we will see shortly, this difference really amounts to the capability of tautology checking.

Example 4.2. ADF $E = (\{a, b\}, \{(b, a), (b, b)\}, \{\varphi_a, \varphi_b\})$ has acceptance formulas $\varphi_a = b \vee \neg b$ and $\varphi_b = \neg b$. So b is self-attacking and the link from b to a is redundant. In Figure 4.1, we show the relevant CPO and the behaviour of the operators $\mathcal{U}_E$ and $\mathcal{G}_E$ on this CPO. ◇

The examples show that the approximate and ultimate families of semantics really are different, save for one straightforward inclusion relation in case of admissible.

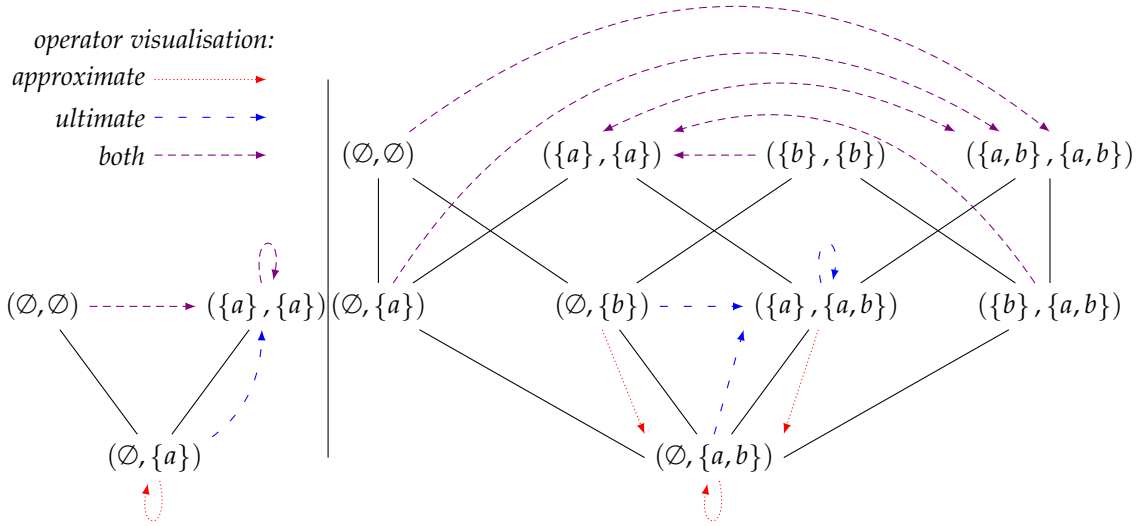


Figure 4.1: Hasse diagrams of consistent CPOs for the ADFs from Example 4.1 (left) and Example 4.2 (right). Solid lines represent the information ordering $\leq_i$. Directed arrows express how revision operators map pairs to other pairs. For pairs where the revisions coincide, the arrows are densely dashed and violet. When the operators revise a pair differently, we use a dotted red arrow for the ultimate and a loosely dashed blue arrow for the approximate operator. Exact (two-valued) pairs are the $\leq_i$-maximal elements. For those pairs, (and any ADF D) it is clear that the operators $\mathcal{U}_D$ and $\mathcal{G}_D$ coincide since they approximate the same two-valued operator G_D. In Example 4.1 on the left, we can see that the ultimate operator maps all pairs to its only fixpoint $(\{a\},\{a\})$ where a is true. The approximate operator has an additional fixpoint, $(\emptyset,\{a\})$, where a is unknown. In Example 4.2 on the right, the major difference between the operators is whether statement a can be derived given that b has truth value unknown. This is the case for the ultimate, but not for the approximate operator. Since there is no fixpoint in the upper row (showing the two-valued operator G_E), the ADF E does not have a two-valued model. Each of the revision operators has however exactly one three-valued fixpoint, which thus constitutes the respective grounded, preferred and complete semantics.

Corollary 4.2. *For any ADF D it holds that an admissible pair of $\mathcal{G}_D$ is an admissible pair of $\mathcal{U}_D$. Let $\sigma \in \{com, grd, pre\}$. There exist ADFs D_1, D_2, D_3 such that:*

1. *there is an admissible pair of $\mathcal{U}_{D_1}$ that is not an admissible pair of $\mathcal{G}_{D_1}$;*
2. *there is a σ-pair of $\mathcal{U}_{D_2}$ that is not a σ-pair of $\mathcal{G}_{D_2}$; and*
3. *there is a σ-pair of $\mathcal{G}_{D_3}$ that is not a σ-pair of $\mathcal{U}_{D_3}$*

Proof. To show that an approximate admissible pair is always an ultimate admissible pair it suffices to consider the fact that $\mathcal{G}_D \leq_i \mathcal{U}_D$. For the remaining claims, we use $D_1 = D_2 = D_3 = E$ from Example 4.2 as a witness:

1. *In Example 4.2, $(\{a\},\{a,b\})$ is ultimate admissible but not approximate admissible.*

2 & 3. *In Example 4.2, we have: (1) approximate grounded, preferred and complete semantics coincide; (2) ultimate grounded, preferred and complete semantics coincide; (3) approximate grounded and ultimate grounded semantics are different with no subset relation either way.* □

4.1.3 Reductions and Encoding Techniques

We now define reductions used in multiple proofs as well as showing some properties of interest. The reductions are defined as functions taking sets of (propositional) variables and a formula and mapping them to an ADF. We generally use the sets P, Q, R for propositional variables and use x, y, z as "gadget" statements in the constructed ADF. Without loss of generality we assume that $\{x, y, z\} \cap (P \cup Q \cup R) = \emptyset$. As usual, links of the ADFs are defined implicitly.

Reduction 4.1. *Let ψ be a propositional formula over the vocabulary P. Define the ADF $\mathsf{RED}_1(P, \psi) = (P \cup \{z\}, L, C)$ with $\varphi_p = \neg p$ for $p \in P$; and $\varphi_z = \psi$.*

It is quite immediate that all $p \in P$ are always undefined for all semantics based on symmetric conflict-freeness.

Lemma 4.3. *Let $D = \mathsf{RED}_1(P, \psi)$ be an ADF obtained from Reduction 4.1 and consider a consistent pair (M, N) that is symmetric conflict-free for D. For each $p \in P$, we find that*

1. *$p \notin M$,*
2. *$p \in N$.*

Proof. Let $p \in P$ and $\mathcal{O} \in \{\mathcal{G}_D, \mathcal{U}_D\}$. Since (M, N) is symmetric conflict-free, we have $M \subseteq \mathcal{O}''(M, N)$ and $\mathcal{O}'(M, N) \subseteq N$.

1. *Assume that $p \in M$. Then $\varphi_p^{(M,N)} = \neg\top \equiv \bot$ and $p \notin \mathcal{O}''(M, N)$. Contradiction.*
2. *Assume that $p \notin N$. Then $\varphi_p^{(M,N)} = \neg\bot \equiv \top$ and $p \in \mathcal{O}'(M, N)$. Contradiction.* □

The simple ADF of Reduction 4.1 can be used to decide satisfiability and refutability of ψ using one of the relevant operators.

Lemma 4.4. *Let ψ be a propositional formula over the vocabulary P and $D = \mathsf{RED}_1(P, \psi)$. Further let $\mathcal{O} \in \{\mathcal{G}_D, \mathcal{U}_D\}$. We find that*

1. *for every symmetric conflict-free pair (X, Y) of $\mathcal{O}$ it holds that $P \subseteq (Y \setminus X)$,*
2. *$z \in \mathcal{U}'_D(\emptyset, P \cup \{z\})$ iff ψ is a tautology,*
3. *$z \in \mathcal{O}''(\emptyset, P \cup \{z\})$ iff ψ is satisfiable,*
4. *$z \notin \mathcal{O}''(\emptyset, P)$ iff ψ is unsatisfiable,*
5. *$(\{z\}, P \cup \{z\})$ is symmetric conflict-free for $\mathcal{O}$ iff ψ is satisfiable,*
6. *$\mathcal{O}(\emptyset, P \cup \{z\}) = \mathcal{O}(\emptyset, P)$.*

Proof. Note that $\mathcal{U}''_D = \mathcal{G}''_D$ by Lemma 4.1.

1. *Follows immediately from Lemma 4.3.*
2. *By definition we have $z \in \mathcal{U}'_D(\emptyset, P \cup \{z\})$ iff for all Z with $\emptyset \subseteq Z \subseteq P \cup \{z\}$ we have $Z \models \varphi_z = \psi$. Clearly if $z \in \mathcal{U}'_D(\emptyset, P \cup \{z\})$, then all two-valued interpretations over P are then satisfying assignments of ψ and ψ is a tautology.*

   For the converse direction assume that ψ is a tautology. Then for all Z with $\emptyset \subseteq Z \subseteq P$ we find $Z \models \psi$. Since z does not occur in $\varphi_z = \psi$ we know that for all Z with $\emptyset \subseteq Z \subseteq P \cup \{z\}$ it holds that $Z \models \psi$ and thus $z \in \mathcal{U}'_D(\emptyset, P \cup \{z\})$.

3. *If $z \in \mathcal{O}''(\emptyset, P \cup \{z\})$, then by definition we can infer that there exists a Z such that $\emptyset \subseteq Z \subseteq P \cup \{z\}$ and $Z \models \psi$. Then $Z \setminus \{z\}$ is a satisfying assignment of ψ, therefore ψ is satisfiable.*

   For the other direction assume that ψ is satisfiable. Then there exists a Z with $\emptyset \subseteq Z \subseteq P$ and $Z \models \psi$. Clearly we have $Z \subseteq P \cup \{z\}$ and thus it holds that $z \in \mathcal{O}''(\emptyset, P \cup \{z\})$.

4. *Follows analogously as the previous item. Note that z is not in the vocabulary of ψ, and if $z \notin \mathcal{O}''(\emptyset, P)$, then by definition we know that for all Z with $\emptyset \subseteq Z \subseteq P$ it holds that $Z \not\models \varphi_z = \psi$.*

   For the other direction, if ψ is unsatisfiable, then for all Z with $\emptyset \subseteq Z \subseteq P$ it holds that $Z \not\models \psi$, and thus $z \notin \mathcal{O}''(\emptyset, P)$.

5. *The pair $(\{z\}, P \cup \{z\})$ is symmetric conflict-free for $\mathcal{O}$ iff $\{z\} \subseteq \mathcal{O}''(\{z\}, P \cup \{z\})$ and $\mathcal{O}'(\{z\}, P \cup \{z\}) \subseteq P \cup \{z\}$. The latter is trivially true. The former holds iff ψ is satisfiable as shown in the third item proven above.*

6. *Lastly, $\mathcal{O}(\emptyset, P \cup \{z\}) = \mathcal{O}(\emptyset, P)$ holds since z does not occur in any acceptance condition.* □

Note that the lemma even implies that $(\{z\}, P \cup \{z\})$ is naive for $\mathcal{O}$ iff ψ is satisfiable, since the elements of P are undecided in any symmetric conflict-free pair.

While the previously introduced reductions are mostly used to show hardness results, we also use reductions for membership results. One general core construction is below.

Reduction 4.2. *Let D be an ADF. Assume that $S = \{s_1, \ldots, s_n\}$ and set $P = \{t_i, u_i, b_{i,j} \mid 1 \leq i, j \leq n\}$. For each statement s_i, the propositional variable t_i indicates that s_i is true, while u_i indicates that s_i is not false. Thus the truth values of the t_i and u_i determine a four-valued interpretation (T, U). The $b_{i,j}$ are used to guess parents that are needed to derive the acceptance of statement s_i in one operator application step; more precisely, $b_{i,j}$ indicates that s_j is a parent of s_i that is "needed" to infer u_i. By φ_i we denote the acceptance formula of s_i; by φ_i^t we denote φ_i where each s_j has been replaced by t_j; by φ_i^b we denote φ_i where each s_j has been replaced by $b_{i,j}$. Now define the formulas (with underlying intuitions on the right)*

$$\phi_{T \subseteq U} = \bigwedge_{s_i \in S} (t_i \to u_i) \qquad (T, U) \text{ is a consistent pair}$$

$$\phi_i^{2\mathsf{v}} = \bigwedge_{r_j \in par(s_i)} (u_j \to t_j) \qquad s_i \text{ has no undecided parents}$$

$$\phi_i^{?} = \bigwedge_{r_j \in par(s_i)} ((t_j \to b_{i,j}) \wedge (b_{i,j} \to u_j)) \qquad \text{guesses for } s_i \text{ are consistent with } (T, U)$$

$$\phi_{\mathsf{fpl}} = \bigwedge_{s_i \in S} (t_i \leftrightarrow (\varphi_i^t \wedge \phi_i^{2\mathsf{v}})) \qquad \mathcal{G}'_D(T, U) = T$$

$$\phi_{\mathsf{fpu}} = \bigwedge_{s_i \in S} (u_i \leftrightarrow (\varphi_i^b \wedge \phi_i^{?})) \qquad \mathcal{G}''_D(T, U) = U$$

$$\phi_{\mathsf{cfp}} = \phi_{\mathsf{fpl}} \wedge \phi_{\mathsf{fpu}} \wedge \phi_{T \subseteq U} \qquad \mathcal{G}_D(T, U) = (T, U) \text{ and } T \subseteq U$$

Finally, set $\mathsf{RED}_3(D) = \phi_{\mathsf{cfp}}$.

The main property of this encoding is that it correctly captures consistent fixpoints of the approximate operator.

Lemma 4.5. *Let D be an ADF over statements S and $\phi_{\mathsf{cfp}} = \mathsf{RED}_3(D)$.*

1. *From each model of ϕ_{cfp}, we can read off a consistent fixpoint of $\mathcal{G}_D$;*
2. *conversely, for each consistent fixpoint of $\mathcal{G}_D$, there is a model of ϕ_{cfp}.*

Proof. 1. *Let $I \subseteq P$ be such that $I \models \phi_{\mathsf{cfp}}$. Define a three-valued pair (T, U) (the associated pair of I) and a sequence $B_1, \ldots, B_n$ by setting*

- *$s_i \in T$ iff $t_i \in I$ and $s_i \in U$ iff $u_i \in I$, and*
- *$s_j \in B_i$ iff $b_{i,j} \in I$.*

We have to show $T \subseteq U$ and $\mathcal{G}_D(T, U) = (T, U)$.

For the first part, let $s_i \in T$. Then $t_i \in I$ by definition. Since $I \models \phi_{\mathsf{cfp}}$, in particular $I \models \phi_{T \subseteq U}$, that is, $I \models \bigwedge_{s_i \in S}(t_i \to u_i)$. Thus $I \models u_i$ and by definition $s_i \in U$.

For the second part, we have

$$\begin{aligned}
s_i \in \mathcal{G}'_D(T, U) \text{ iff } & T \models \varphi_i \text{ and } par(s_i) \cap U \subseteq par(s_i) \cap T \\
\text{iff } & I \models \varphi_i^t \text{ and } I \models \phi_i^{\mathsf{2v}} \\
\text{iff } & I \models \varphi_i^t \wedge \phi_i^{\mathsf{2v}} \\
\text{iff } & I \models t_i && \text{(since } I \models \phi_{\mathsf{fpl}}\text{)} \\
\text{iff } & s_i \in T
\end{aligned}$$

Hence $\mathcal{G}'_D(T, U) = T$. Similarly, for the upper bound we have

$$\begin{aligned}
s_i \in \mathcal{G}'_D(U, T) \text{ iff } & B_i \models \varphi_i \text{ and } par(s_i) \setminus B_i \subseteq S \setminus T \text{ and } B_i \subseteq U \\
\text{iff } & I \models \varphi_i^b \text{ and } I \models \bigwedge_{s_j \in S} ((\neg b_{i,j} \to \neg t_j) \wedge (b_{i,j} \to u_j)) \\
\text{iff } & I \models \varphi_i^b \text{ and } I \models \bigwedge_{s_j \in S} ((t_j \to b_{i,j}) \wedge (b_{i,j} \to u_j)) \\
\text{iff } & I \models \varphi_i^b \text{ and } I \models \phi_i^? \\
\text{iff } & I \models (\varphi_i^b \wedge \phi_i^?) && \text{(since } I \models \phi_{\mathsf{fpu}}\text{)} \\
\text{iff } & I \models u_i \\
\text{iff } & s_i \in U
\end{aligned}$$

Hence $U = \mathcal{G}'_D(U, T)$ and in combination $\mathcal{G}_D(T, U) = (T, U)$.

2. *Let $\mathcal{G}_D(T, U) = (T, U)$ with $T \subseteq U$. Define an interpretation $I \subseteq P$ as follows:*
   - *Set $t_i \in I$ iff $s_i \in T$ and $u_i \in I$ iff $s_i \in U$.*
   - *Since $\mathcal{G}'_D(U, T) = U$, we have for each $1 \leq i \leq n$ that $s_i \in U$ iff there is a $B_i \subseteq par(s_i)$ with $B_i \models \varphi_i$, $par(s_i) \setminus B_i \subseteq S \setminus T$ and $B_i \subseteq U$. Now pick such a B_i for each $s_i \in S$ and set $b_{i,j} \in I$ iff $s_j \in B_i$.*

We have to show $I \models \phi_{\mathsf{cfp}}$. *Since* $T \subseteq U$, *it is clear that* $I \models \phi_{T \subseteq U}$ *since for all* i, $s_i \in T$ *implies* $s_i \in U$. *For the operator applications, we get, for any* $s_i \in S$,

$$\begin{aligned} I \models t_i \text{ iff } & s_i \in T \\ & \text{iff } s_i \in \mathcal{G}'_D(T, U) \\ & \text{iff } T \models \varphi_i \text{ and } par(s_i) \cap U \subseteq par(s_i) \cap T \\ & \text{iff } I \models \varphi_i^t \text{ and } I \models \phi_i^{\mathsf{2v}} \\ & \text{iff } I \models \varphi_i^t \wedge \phi_i^{\mathsf{2v}} \end{aligned}$$

Thus $I \models \phi_{\mathsf{fpl}}$. *For the upper bound, for any* $s_i \in S$,

$$\begin{aligned} I \models u_i \text{ iff } & s_i \in U \\ & \text{iff } s_i \in \mathcal{G}'_D(U, T) \\ & \text{iff } B_i \models \varphi_i \text{ and } par(s_i) \setminus B_i \subseteq S \setminus T \text{ and } B_i \subseteq U \\ & \text{iff } I \models \varphi_i^b \text{ and } I \models \bigwedge_{s_j \in S} ((\neg b_{i,j} \rightarrow \neg t_j) \wedge (b_{i,j} \rightarrow u_j)) \\ & \text{iff } I \models \varphi_i^b \text{ and } I \models \bigwedge_{s_j \in S} ((t_j \rightarrow b_{i,j}) \wedge (b_{i,j} \rightarrow u_j)) \\ & \text{iff } I \models \varphi_i^b \text{ and } I \models \phi_i^? \end{aligned}$$

Hence $I \models \phi_{\mathsf{fpu}}$ *and in total* $I \models \phi_{\mathsf{fpl}} \wedge \phi_{\mathsf{fpu}} \wedge \phi_{T \subseteq U}$. □

4.1.4 Operator Complexities

We next analyse the computational complexity of deciding whether a single statement is contained in the lower or upper bound of the revision of a given pair. This then leads to the complexity of checking whether current lower/upper bounds are pre- or postfixpoints of the revision operators for computing new lower/upper bounds, that is, whether the revisions represent improvements in terms of the information ordering. Intuitively, these results describe how hard it is to "use" the operators and lay the foundation for the rest of the complexity results. Formally we express these notions via the decision problems $\mathsf{Elem}^{\mathcal{I}'}$ and $\mathsf{RevBound}^{\mathcal{I}'}_{\circ}$ with $\circ \in \{\subseteq, \supseteq\}$, respectively with $\mathcal{I}''$ in the superscript. Recall that $\mathsf{Elem}^{\mathcal{I}'}$ ($\mathsf{Elem}^{\mathcal{I}''}$) denotes the decision problem of verifying if a given element (statement) is contained in the revision of the lower (upper) bound of a given operator and pair. The problem $\mathsf{RevBound}^{\mathcal{I}'}_{\circ}$ asks whether for a given pair (X, Y) we can compare a given set $B \subseteq S$ via $\circ$ with the revised lower bound of this pair. For instance $\mathsf{RevBound}^{\mathcal{G}'}_{\supseteq}$ denotes the problem of verifying that for a given (X, Y), $B \subseteq S$ and $\mathcal{G}_D \in \mathcal{G}$ we have $\mathcal{G}'_D(X, Y) \subseteq B$.

Proposition 4.6. *Let* $\mathcal{I} \in \{\mathcal{G}, \mathcal{U}\}$. *It holds that*

1. $\mathsf{Elem}^{\mathcal{G}'}$ *is in* P,
2. $\mathsf{Elem}^{\mathcal{U}'}$ *is* coNP*-complete,*
3. $\mathsf{Elem}^{\mathcal{I}''}$ *is* NP*-complete.*

Proof. Let D be an ADF, $s \in S$ and $X \subseteq Y \subseteq S$.

1. $\mathsf{Elem}^{\mathcal{G}'}$ *is in* P*: Since $X \subseteq Y$, we have that whenever there exists a $B \subseteq X \cap par(s)$ with $C_s(B) = \mathbf{t}$ and $par(s) \setminus B \subseteq S \setminus Y$, we know that $B = X \cap par(s)$: Assume there is an $r \in (X \cap par(s)) \setminus B$. Then $r \in par(s)$ and $r \notin B$, whence $r \in par(s) \setminus B \subseteq S \setminus Y$. By $r \in X \subseteq Y$ we get $r \notin S \setminus Y$, contradiction. Thus $B = X \cap par(s)$. Now*

$$\begin{aligned} s \in \mathcal{G}'_D(X,Y) \text{ iff there exists } B \subseteq X \cap par(s) \text{ with } C_s(B) = \mathbf{t} \text{ and } par(s) \setminus B \subseteq S \setminus Y \\ \text{iff } C_s(X \cap par(s)) = \mathbf{t} \text{ and } par(s) \setminus X \subseteq S \setminus Y \\ \text{iff } C_s(X \cap par(s)) = \mathbf{t} \text{ and } (Y \setminus X) \cap par(s) = \emptyset \end{aligned}$$

   For acceptance functions represented by propositional formulas, $C_s(X \cap par(s)) = \mathbf{t}$ can be decided in polynomial time, since we only have to check whether $X \models \varphi_s$. It can be decided in quadratic time whether there is an undecided parent $r \in par(s)$ with $r \in Y \setminus X$.

2. $\mathsf{Elem}^{\mathcal{U}'}$ *is* coNP*-complete:*

   in coNP: *To decide that $s \notin \mathcal{U}'_D(X,Y)$, we guess a Z with $X \subseteq Z \subseteq Y$ and verify that $Z \not\models \varphi_s$.*

   coNP-hard: *We provide a reduction from the problem of determining if a given propositional formula ψ is a tautology. Let ψ be an arbitrary formula over vocabulary P. Construct $D_\psi = \mathsf{RED}_1(P,\psi)$ as defined in Reduction 4.1. By Lemma 4.4 it follows that $z \in \mathcal{U}'_{D_\psi}(\emptyset, P \cup \{z\})$ iff ψ is a tautology.*

3. $\mathsf{Elem}^{\mathcal{I}''}$ *is* NP*-complete: Due to Lemma 4.1 we know that $\mathcal{G}''_D(X,Y) = \mathcal{U}''_D(X,Y)$.*

   in NP: *To decide that $s \in \mathcal{U}''_D(X,Y)$, we guess a Z with $X \subseteq Z \subseteq Y$ and verify that $Z \models \varphi_s$.*

   NP-hard: *For hardness, we provide a reduction from SAT. Let ψ be a propositional formula over vocabulary P. Construct $D_\psi = \mathsf{RED}_1(P,\psi)$ as defined in Reduction 4.1. By Lemma 4.4 it follows that $z \in \mathcal{U}''_{D_\psi}(\emptyset, P \cup \{z\})$ iff ψ is satisfiable.* □

These results can also be formulated in terms of partial evaluations of acceptance formulas:

- We have $s \in \mathcal{G}'_D(X,Y)$ iff the partial evaluation $\varphi_s^{(X,Y)}$ is a formula without variables that evaluates to $\mathbf{t}$.
- Similarly, we have $s \in \mathcal{G}''_D(X,Y) = \mathcal{U}''_D(X,Y)$ iff the partial evaluation $\varphi_s^{(X,Y)}$ is satisfiable.

Under standard complexity assumptions, computing a new lower bound with the ultimate operator is harder than with the approximate operator. This is because, intuitively, $s \in \mathcal{U}'_D(X,Y)$ iff the partial evaluation $\varphi_s^{(X,Y)}$ is a tautology. The results for Elem straightforwardly lead to the complexity of revising lower/upper bounds for both operators. Note that the results depend crucially on restricting revision to *consistent* pairs (X,Y) (those with $X \subseteq Y$) – for otherwise we could apply $\mathcal{G}''_D(X,Y) = \mathcal{G}'_D(Y,X)$ and use the polynomial-time computable approximate lower bound operator $\mathcal{G}'_D$ on an inconsistent pair (Y,X) to compute $\mathcal{G}'_D(Y,X) = \mathcal{G}''_D(X,Y)$.

Lemma 4.7. *Let $\mathcal{I} \in \{\mathcal{G}, \mathcal{U}\}$ and $B \in \{L, U\}$. It holds that*

1. $\mathsf{RevBound}^{\mathcal{G}'}_{\supseteq}$ *and* $\mathsf{RevBound}^{\mathcal{G}'}_{\subseteq}$ *are in* P*,*
2. $\mathsf{RevBound}^{\mathcal{U}'}_{\supseteq}$ *is in* NP*,*

3. $\mathsf{RevBound}_{\subseteq}^{\mathcal{U}'}$ *is in* coNP,

4. $\mathsf{RevBound}_{\subseteq}^{\mathcal{I}''}$ *is in* NP,

5. $\mathsf{RevBound}_{\supseteq}^{\mathcal{I}''}$ *is in* coNP.

Proof. All results build upon Proposition 4.6. Since the revised lower bound w.r.t. $\mathcal{G}_D$ can be computed in polynomial time for any ADF D we can immediately infer the complexity of the corresponding problems $\mathsf{RevBound}_{\supseteq}^{\mathcal{G}'}$ *and* $\mathsf{RevBound}_{\subseteq}^{\mathcal{G}'}$.

Let $D = (S, L, C)$ be an ADF, $\mathcal{O} \in \{\mathcal{G}_D, \mathcal{U}_D\}$, $B \subseteq S$ and $X \subseteq Y \subseteq S$. Deciding whether $B \subseteq \mathcal{U}'_D(X, Y)$ can be decided via $|B|$ independent checks for each $b \in B$ whether $b \in \mathcal{U}'_D(X, Y)$. Each of these are checks in coNP *and combining them yields again a check in* coNP. *Therefore* $\mathsf{RevBound}_{\subseteq}^{\mathcal{U}'}$ *is in* coNP. *Likewise deciding whether $B \subseteq \mathcal{O}''(X, Y)$ can be decided via $|B|$ independent checks $b \in B$, each of them in* NP, *yielding again a combined problem in* NP. *Thus* $\mathsf{RevBound}_{\subseteq}^{\mathcal{I}''}$ *is in* NP.

For $\mathcal{U}'_D(X, Y) \subseteq B$ we can decide for each $s \in (S \setminus B)$ whether $s \notin \mathcal{U}'_D(X, Y)$. If this is the case for all s, then it holds that $\mathcal{U}'_D(X, Y) \subseteq B$. Deciding whether $s \notin \mathcal{U}'_D(X, Y)$ holds is a complementary problem to one in coNP, *thus combining several of them yields a problem in* NP. *This directly shows that* $\mathsf{RevBound}_{\supseteq}^{\mathcal{U}'}$ *is in* NP. *The proof that* $\mathsf{RevBound}_{\supseteq}^{\mathcal{I}''}$ *is in* coNP *proceeds analogously.* □

4.1.5 Generic Upper Bounds

We now show generic upper bounds for the computational complexity of the considered problems. This kind of analysis is in the spirit of the results by Dimopoulos et al. (2002, Section 4). The first item is furthermore a straightforward generalisation of Theorem 6.13 by Denecker et al. (2004).

Theorem 4.8. *Let $\mathcal{I} \subseteq \mathcal{A}$ be a set of approximation operators, each defined on a CPO on S-subset pairs for some finite set S. Further let* $\mathsf{Elem}^{\mathcal{I}'}$ *be in Π_i^P and* $\mathsf{Elem}^{\mathcal{I}''}$ *be in Σ_i^P.*

1. *The least fixpoint of an $\mathcal{O} \in \mathcal{I}$ can be computed in polynomial time with a polynomial number of calls to a Σ_i^P-oracle.*

2. $\mathsf{Ver}_{cfi}^{\mathcal{I}}$ *is in Σ_i^P;* $\mathsf{Cred}_{cfi}^{\mathcal{I}}$ *is in Σ_i^P;*

3. $\mathsf{Ver}_{nai}^{\mathcal{I}}$ *is in D_i^P;* $\mathsf{Cred}_{nai}^{\mathcal{I}}$ *is in Σ_i^P;*

4. $\mathsf{Ver}_{adm}^{\mathcal{I}}$ *is in Π_i^P;* $\mathsf{Cred}_{adm}^{\mathcal{I}}$ *is in Σ_{i+1}^P;*

5. $\mathsf{Ver}_{com}^{\mathcal{I}}$ *is in D_i^P;* $\mathsf{Cred}_{com}^{\mathcal{I}}$ *is in Σ_{i+1}^P;*

6. $\mathsf{Ver}_{pre}^{\mathcal{I}}$ *is in Π_{i+1}^P;* $\mathsf{Cred}_{pre}^{\mathcal{I}}$ *is in Σ_{i+1}^P;* $\mathsf{Scep}_{pre}^{\mathcal{I}}$ *is in Π_{i+2}^P.*

Proof. Let $A = 2^S$ and $\mathcal{O}$ be an approximating operator on $(A^c, \leq_i)$, the consistent CPO of S-subset pairs. Further let $(X, Y) \in A^c$ and $s \in S$.

Using the same line of reasoning as in the proof of Lemma 4.7 we can immediately infer that under the assumptions of the current theorem that $\mathsf{RevBound}_{\supseteq}^{\mathcal{I}'}$ *is in Σ_i^P,* $\mathsf{RevBound}_{\subseteq}^{\mathcal{I}'}$ *is in Π_i^P,* $\mathsf{RevBound}_{\subseteq}^{\mathcal{I}''}$ *is in Σ_i^P,* $\mathsf{RevBound}_{\supseteq}^{\mathcal{I}''}$ *is in Π_i^P.*

1. *For any $(V, W) \in A^c$ we can use the oracle to compute an application of $\mathcal{O}'$ by simply asking whether $z \in \mathcal{O}'(V, W)$ for each $z \in S$. This means we can compute with a linear number of oracle calls the sets $\mathcal{O}'(V, W)$ and $\mathcal{O}''(V, W)$, thus the pair $\mathcal{O}(V, W)$. Hence we can compute the sequence $(\emptyset, S) \leq_i \mathcal{O}(\emptyset, S) \leq_i \mathcal{O}(\mathcal{O}(\emptyset, S)) \leq_i \ldots$ which converges to the least fixpoint of $\mathcal{O}$ after a linear number of operator applications (and thus a polynomial number of oracle calls).*

2. $\mathsf{Ver}^{\mathcal{I}}_{cfi}$ *is in* Σ^P_i*, since we can verify if a given pair is symmetric conflict-free for a given operator if the lower bound is a positive instance of* $\mathsf{RevBound}^{\mathcal{I}''}_{\subseteq}$ *and the upper bound a positive instance* $\mathsf{RevBound}^{\mathcal{I}'}_{\supseteq}$ *together with the pair and operator as input. These are two independent checks in* Σ^P_i*. For* $\mathsf{Cred}^{\mathcal{I}}_{cfi}$ *it suffices to verify that the pair* $(\{s\}, S)$ *is symmetric conflict-free. (If* $(\{s\}, S)$ *is not symmetric conflict-free, by Lemma 3.48 there is no symmetric conflict-free pair* (X, Y) *with* $(\{s\}, S) \leq_i (X, Y)$*.)*

3. *For* $\mathsf{Ver}^{\mathcal{I}}_{nai}$ *we first have to decide* $\mathsf{Ver}^{\mathcal{I}}_{cfi}$*, which can be done in* Σ^P_i*. To verify that* (X, Y) *is naive, that is,* $\leq_i$*-maximal, we do the following: Assume that* $Y \setminus X = \{s_1, \ldots, s_m\}$ *and construct the pairs* $\bar{p}_i = (X \cup \{s_i\}, Y)$ *and* $\bar{q}_i = (X, Y \setminus \{s_i\})$ *for* $1 \leq i \leq m$*. It follows from Lemma 3.48 that the pair* (X, Y) *is naive iff* none *of the* $2m$ *pairs* $\bar{p}_1, \ldots, \bar{p}_m, \bar{q}_1, \ldots, \bar{q}_m$ *is symmetric conflict-free. Since* $\mathsf{Ver}^{\mathcal{I}}_{cfi}$ *is in* Σ^P_i *and the pairs can be verified independently of each other, we need to solve at most* $2m \leq 2 \cdot |S|$ *independent* Σ^P_i *problems to show that* (X, Y) *is* not *naive. Thus showing that* (X, Y) *is* $\leq_i$*-maximal is in* Π^P_i*, and together with showing symmetric conflict-freeness of* (X, Y) *in* Σ^P_i *the containment in* D^P_i *follows.* $\mathsf{Cred}^{\mathcal{I}}_{nai}$ *coincides with* $\mathsf{Cred}^{\mathcal{I}}_{cfi}$ *by Lemma 4.13.*

4. $\mathsf{Ver}^{\mathcal{I}}_{adm}$ *is in* Π^P_i*, since we can verify if a given pair is admissible for a given operator if the lower bound is a positive instance of* $\mathsf{RevBound}^{\mathcal{I}'}_{\subseteq}$ *and the upper bound a positive instance* $\mathsf{RevBound}^{\mathcal{I}''}_{\supseteq}$ *together with the pair and operator as input. These are two independent checks in* Π^P_i*. For* $\mathsf{Cred}^{\mathcal{I}}_{adm}$*, we guess a pair* (X_1, Y_1) *with* $s \in X_1$ *and check if it is admissible.*

5. $\mathsf{Ver}^{\mathcal{I}}_{com}$ *is in* D^P_i*, since we can verify if a given pair is admissible for a given operator in* Π^P_i*. By determining if the lower bound is a positive instance of* $\mathsf{RevBound}^{\mathcal{I}'}_{\supseteq}$ *and the upper bound a positive instance of* $\mathsf{RevBound}^{\mathcal{I}''}_{\subseteq}$ *together with the pair and operator as input we can infer that the given pair is a fix point of the operator. These are two independent checks in* Σ^P_i*, thus combined yields a check in* D^P_i*.* $\mathsf{Cred}^{\mathcal{I}}_{com} = \mathsf{Cred}^{\mathcal{I}}_{adm}$ *by Lemma 4.26.*

6. *For* $\mathsf{Ver}^{\mathcal{I}}_{pre}$*, we show that the co-problem is in* Σ^P_{i+1}*. To show that* (X, Y) *is not a preferred pair, we can show that (1)* (X, Y) *is not a complete pair, which can be decided in* D^P_i*; or (2) that there is a complete pair* (X_1, Y_1) *with* $(X, Y) <_i (X_1, Y_1)$*, which can be done by guessing* (X_1, Y_1) *and showing in* D^P_i *that* (X_1, Y_1) *is complete.*

   $\mathsf{Cred}^{\mathcal{I}}_{pre}$*: coincides with credulous reasoning w.r.t. admissibility, see Lemma 4.26.*

   $\mathsf{Scep}^{\mathcal{I}}_{pre}$*: Consider the co-problem, i.e. deciding whether there exists a preferred pair* (X_1, Y_1) *with* $X_1 \cap \{a\} = \emptyset$*. We guess such a pair* (X_1, Y_1) *and check if it is preferred.* □

Naturally, the capability of solving the functional problem of *computing* the grounded semantics allows us to solve the associated decision problems.

Corollary 4.9. *Under the assumptions of Theorem 4.8, the problems* $\mathsf{Ver}^{\mathcal{I}}_{grd}$ *and* $\mathsf{Exists}^{\mathcal{I}}_{grd}$ *are in* Δ^P_{i+1}*.*

4.2 General ADFs

Proposition 4.10. *Deciding whether $D = (S, L, C)$ has a two-valued model is* NP*-complete.*

Proof. For membership, we can guess an interpretation $v : S \to \{\mathbf{t}, \mathbf{f}\}$ and in polynomial time check whether $v(s) = v(\varphi_s)$ for each $s \in S$.

For hardness, let ψ be a propositional formula over a vocabulary P. We construct an ADF D_ψ that has a model iff ψ is satisfiable. Set $S = P \cup \{s\}$, $L = S \times S$ and for the acceptance formulas set $\varphi_p = p$ for each $p \in P$ and $\varphi_s = \neg s \wedge \neg\psi$. We have to show that D_ψ has a model iff ψ is satisfiable. If ψ is satisfiable, there exists a satisfying valuation v for P. Then $v(\neg\psi) = \mathbf{f}$ and $v(s) = \mathbf{f} = v(\varphi_s)$ and v is a model for D_ψ. Now let ψ be unsatisfiable and assume to the contrary that D_ψ has a model v. Obviously $v(\neg\psi) = \mathbf{t}$ and thus $v(s) = v(\varphi_s) = v(\neg s \wedge \neg\psi) = v(\neg s)$, contradiction. □

4.2.1 Symmetric Conflict-free Semantics

For an ADF D and an operator $\mathcal{O} \in \{\mathcal{G}_D, \mathcal{U}_D\}$, a pair (X, Y) is symmetric conflict-free by definition if and only if $X \subseteq \mathcal{O}''(X, Y)$ and $\mathcal{O}'(X, Y) \subseteq Y$. For the ultimate operator $\mathcal{U}_D$, this intuitively means the following:

- For every statement $s \in X$ that is set to true, its partially evaluated acceptance formula $\varphi_s^{(X,Y)}$ must be satisfiable.
- For every statement $s \in S \setminus Y$ that is set to false, its partially evaluated acceptance formula $\varphi_s^{(X,Y)}$ must be refutable.

So roughly, symmetric conflict-freeness dictates that the pair must not make truth value assignments that are completely absurd in that a statement is set to true in the pair although its acceptance formula is unsatisfiable with respect to the pair (or symmetrically set to false while the formula is a tautology).

For the approximate operator $\mathcal{G}_D$, the requirement for setting statements to false is weaker than for the ultimate operator. (The requirement for setting statements to true is the same since $\mathcal{G}_D'' = \mathcal{U}_D''$.) For the approximate operator, a statement $s \in S$ can be set to false in a pair (X, Y) as long as it is not the case that the formula $\varphi_s^{(X,Y)}$ is a Boolean expression consisting of truth values and connectives that evaluates to true. Conversely, the statement can be set to false if either (1) the formula $\varphi_s^{(X,Y)}$ is a Boolean expression consisting of truth values and connectives that evaluates to false, or (2) the formula $\varphi_s^{(X,Y)}$ contains variables.

Example 4.3. Consider the ADF $D = (S, L, C)$ with $S = \{a, b\}$ and L and C given by $\varphi_a = \neg a$ and $\varphi_b = a \vee \neg a$. For any pair (X, Y) with $a \in X$, that is, any pair that sets a to true, we have that $\varphi_a^{(X,Y)} = \neg\top \equiv \bot$ is unsatisfiable. Thus such a pair is not ultimate symmetric conflict-free. Symmetrically, for any pair (X, Y) with $a \notin Y$, we find that $\varphi_a^{(X,Y)} = \neg\bot \equiv \top$ is irrefutable, and the pair is also not ultimate symmetric conflict-free. So our only chance is to set a to undecided, that is, $a \in Y$ and $a \notin X$. For statement b, we see that $\varphi_b^{(\{b\},\{a,b\})} = a \vee \neg a$ is satisfiable, whence $(\{b\}, \{a, b\})$ is ultimate symmetric conflict-free. For the pair $(\emptyset, \{a\})$ where b is false, we see that $\varphi_b^{(\emptyset,\{a\})} = a \vee \neg a$ is a tautology, whence the pair is not an ultimate symmetric conflict-free pair. However, $\varphi_b^{(\emptyset,\{a\})} = a \vee \neg a$ is an expression containing variables, whence $(\emptyset, \{a\})$ is an approximate symmetric conflict-free pair. ◇

This intuition based on satisfiability and refutability will help us in obtaining complexity results for semantics based on the property of being symmetric conflict-free. To begin with, to verify that a pair is symmetric conflict-free, we obviously have to solve a combined satisfiability/refutability problem.

Proposition 4.11. *Consider any $\mathcal{I} \in \{\mathcal{G}, \mathcal{U}\}$. $\mathsf{Ver}^{\mathcal{I}}_{scf}$ is NP-complete.*

Proof. Membership follows from Theorem 4.8. For hardness, we provide a reduction from SAT by Reduction 4.1. Let ψ be a formula over vocabulary P. Let $D = \mathsf{RED}_1(P, \psi)$ and $\mathcal{O} \in \{\mathcal{G}_D, \mathcal{U}_D\}$. Due to Lemma 4.4 we have that the pair $(\{z\}, P \cup \{z\})$ is symmetric conflict-free for $\mathcal{O}$ iff ψ is satisfiable.□

For deciding whether there exists a non-trivial ultimate symmetric conflict-free pair, we can reduce the propositional satisfiability problem back and forth.

Proposition 4.12. $\mathsf{Exists}^{\mathcal{U}}_{scf}$ *is* NP*-complete.*

Proof. in NP: *We can guess a non-trivial pair (X, Y) and the witnesses that verify conflict-freeness of (X, Y) in one sequence of independent guesses. More formally, we reduce $\mathsf{Exists}^{\mathcal{U}}_{scf}$ to SAT. Let $D = (S, L, C)$ be an ADF. We define a formula φ_D such that $\mathcal{U}_D$ has a non-trivial conflict-free pair iff φ_D is satisfiable.*

Assume $S = \{s_1, \ldots, s_n\}$ and define the vocabulary of φ_D as

$$P = \left\{ s_i^{\mathbf{t}}, s_i^{\mathbf{f}}, p_{i,j} \;\middle|\; 1 \leq i, j \leq n \right\}$$

For $1 \leq i, j \leq n$, denote by φ_i the acceptance formula φ_{s_i} where each occurrence of s_j has been replaced by $p_{j,i}$. Intuitively, atom $p_{j,i}$ is used to guess a truth value for s_j in the acceptance formula of s_i. Now define

$$\varphi_{scf}^{\mathbf{t},i} = s_i^{\mathbf{t}} \rightarrow \left(\varphi_i \wedge \bigwedge_{1 \leq j \leq n} p_{i,j} \right)$$

$$\varphi_{scf}^{\mathbf{f},i} = s_i^{\mathbf{f}} \rightarrow \left(\neg\varphi_i \wedge \bigwedge_{1 \leq j \leq n} \neg p_{i,j} \right)$$

$$\varphi_{scf} = \bigwedge_{1 \leq i \leq n} \left(\varphi_{scf}^{\mathbf{t},i} \wedge \varphi_{scf}^{\mathbf{f},i} \right)$$

$$\varphi_{\mathsf{nt}} = \bigvee_{1 \leq i \leq n} \left(s_i^{\mathbf{t}} \vee s_i^{\mathbf{f}} \right)$$

$$\varphi_D = \varphi_{scf} \wedge \varphi_{\mathsf{nt}}$$

We have to show that $\mathcal{U}_D$ has a non-trivial conflict-free pair iff φ_D is satisfiable.

if: *Let $I \subseteq P$ be a model for φ_D. Define the pair (X, Y) by*

$$X = \left\{ s_i \;\middle|\; s_i^{\mathbf{t}} \in I, 1 \leq i \leq n \right\}$$

$$Y = \left\{ s_i \;\middle|\; s_i^{\mathbf{f}} \notin I, 1 \leq i \leq n \right\}$$

Since in particular I is a model for φ_{nt}, there is an $i \in \{1, \ldots, n\}$ such that $s_i \in X$ or $s_i \notin Y$, that is, (X, Y) is non-trivial. It remains to show that (X, Y) is conflict-free.

$X \subseteq \mathcal{U}''_D(X,Y)$: *Let $s_i \in X$. By definition $s_i^{\mathbf{t}} \in I$ and thus I is a model for φ_i. Define $J = \{s_j \mid p_{j,i} \in I\}$. For each $1 \leq k \leq n$, we have that $s_k \in X$ implies $s_k^{\mathbf{t}} \in I$ and thus $p_{k,i} \in I$, whence $s_k \in J$; likewise $s_k \notin Y$ implies $s_k^{\mathbf{f}} \in I$ and thus $p_{k,i} \notin I$ whence $s_k \notin J$. Now I is a model for φ_i, thus J is a model for $\varphi_{s_i}^{(X,Y)}$.*

$\mathcal{U}'_D(X,Y) \subseteq Y$: *Let $s_i \in S \setminus Y$. By definition $s_i^{\mathbf{f}} \in I$ and I falsifies φ_i. As above, we can define J and show that it is compatible with (X,Y). Consequently, J falsifies $\varphi_{s_i}^{(X,Y)}$.*

only if: *Let (X,Y) be a non-trivial conflict-free pair. Define an interpretation I of P as follows. For each $s_i \in X$ set $s_i^{\mathbf{t}}$ to true, $s_i^{\mathbf{f}}$ to false, and $p_{i,j}$ to true for all $1 \leq j \leq n$; for each $s_i \in S \setminus Y$ set $s_i^{\mathbf{f}}$ to true, $s_i^{\mathbf{t}}$ to false, and $p_{i,j}$ to false for all $1 \leq j \leq n$. (There exists at least one such s_i since (X,Y) is non-trivial.) Now let $s_k \in S$.*

- *If $s_k \in X$, we have $s_k \in \mathcal{U}''_D(X,Y)$, whence $\varphi_{s_k}^{(X,Y)}$ is satisfiable. Let I_k be a model for $\varphi_{s_k}^{(X,Y)}$ and note that $I_k \subseteq Y \setminus X$. Now for each $s_j \in Y \setminus X$, set $p_{j,k} \in I$ iff $s_j \in I_k$. Clearly I satisfies φ_k by construction. Since we already defined $p_{k,j}$ to be true for all $1 \leq j \leq n$, we find that I is a model for $\varphi_{scf}^{\mathbf{t},k}$. Since $s_k^{\mathbf{f}}$ is false, I is also a model for $\varphi_{scf}^{\mathbf{f},k}$.*
- *If $s_k \in S \setminus Y$, we have $s_k \notin \mathcal{U}'_D(X,Y)$, whence $\varphi_{s_k}^{(X,Y)}$ is refutable. As above, let I_k be a falsification of $\varphi_{s_k}^{(X,Y)}$ and for $s_j \in Y \setminus X$ set $p_{j,k} \in I$ iff $s_j \in I_k$. Then I falsifies φ_k. Again, I is a model for $\varphi_{scf}^{\mathbf{f},k}$ and $\varphi_{scf}^{\mathbf{t},k}$.*
- *If $s_k \in Y \setminus X$, we set $s_k^{\mathbf{t}}$ and $s_k^{\mathbf{f}}$ to false in I and for $s_j \in Y \setminus X$ the $p_{j,k}$ arbitrary in I. Clearly I is a model for $\varphi_{scf}^{\mathbf{t},k}$ and $\varphi_{scf}^{\mathbf{f},k}$.*

Hence I is a model for φ_{scf}; since (X,Y) is non-trivial, I is also a model of φ_{nt}. Thus I is a model for φ_D.

NP-hard: *We provide a reduction from SAT. Let ψ be a propositional formula over vocabulary $P \neq \emptyset$. Define an ADF $D_\psi = (S, L, C)$ with $S = P \cup \{z\}$ (where $z \notin P$), $\varphi_p = \neg p$ for $p \in P$ and $\varphi_z = \neg z \wedge \neg\psi$. We have to show that $\mathcal{U}_{D_\psi}$ has a non-trivial conflict-free pair iff ψ is satisfiable.*

if: *If ψ is satisfiable, then $\neg\psi$ is refutable and $\varphi_z^{(\emptyset,P)} = \neg\bot \wedge \neg\psi$ is refutable. Thus $z \notin \mathcal{U}'_{D_\psi}(\emptyset,P)$, whence $\emptyset \subseteq \mathcal{U}''_{D_\psi}(\emptyset,P)$ and $\mathcal{U}'_{D_\psi}(\emptyset,P) \subseteq P$ and $(\emptyset,P)$ is non-trivial ($z \notin P$) and conflict-free.*

only if: *Let (X,Y) be conflict-free and non-trivial. Then $X \neq \emptyset$ or $Y \subsetneq S$. Clearly conflict-freeness implies that $P \cap X = \emptyset$ and $P \subseteq Y$. So $z \in X$ or $z \notin Y$. If $z \in X$ then $(X,Y) = (\{z\}, S)$ and by conflict-freeness $z \in \mathcal{U}''_{D_\psi}(\{z\}, S)$. Then $\varphi_z^{(\{z\},S)} = \neg\top \wedge \neg\psi$ is satisfiable, contradiction. Thus $X = \emptyset$ and $z \notin Y$, that is, $(X,Y) = (\emptyset,P)$. Since $(\emptyset,P)$ is conflict-free, $z \notin \mathcal{U}'_{D_\psi}(\emptyset,P)$. Thus $\varphi_z^{(\emptyset,P)} = \neg\bot \wedge \neg\psi$ is refutable, that is, $\neg\psi$ is refutable whence ψ is satisfiable.* □

Now one of the general existence results of Section 3.6 becomes useful: as an easy consequence of Theorem 3.47, the existence of non-trivial (symmetric) naive pairs is then equivalent to the existence of non-trivial symmetric conflict-free pairs.

Lemma 4.13. *Let $(L, \sqsubseteq)$ be a complete lattice and $\mathcal{O}$ an approximating operator on $(L^c, \leq_i)$. The following are equivalent:*

1. *$\mathcal{O}$ has a non-trivial conflict-free pair.*
2. *$\mathcal{O}$ has a non-trivial naive pair.*

Proof. "(1) $\Longrightarrow$ (2)": *Let (x,y) be non-trivial and conflict-free, that is, in particular let $(\bot_L, \top_L) <_i (x,y)$. By Theorem 3.47, there exists a naive pair $(p,q) \in L^c$ with $(\bot_L, \top_L) <_i (x,y) \leq_i (p,q)$.*

"(2) $\Longrightarrow$ (1)": *Any naive pair is conflict-free (Definition 3.15).* □

This directly shows the equivalence of the respective decision problems, that is, it holds that $\mathsf{Exists}^{\mathcal{A}}_{cfi} = \mathsf{Exists}^{\mathcal{A}}_{nai}$.

Corollary 4.14. $\mathsf{Exists}^{\mathcal{U}}_{nai}$ *is* NP*-complete.*

Fortunately, credulous reasoning over conflict-free pairs is not harder than just guessing a pair where the desired statement is true.

Proposition 4.15. *Consider any $\mathcal{I} \in \{\mathcal{G}, \mathcal{U}\}$.* $\mathsf{Cred}^{\mathcal{I}}_{scf}$ *is* NP*-complete.*

Proof. in NP: *Let $D = (S, L, C)$ be an ADF with $S = \{s_1, \ldots, s_n\}$ and $1 \leq k \leq n$. Intuitively, we can guess a pair (X,Y) with $s_k \in X$ along with the witnesses showing that (X,Y) is conflict-free. More formally, we reduce the problem to SAT. For the ultimate operator, we can adapt the construction of Proposition 4.12. We use the formula φ_{scf} as above and define $\varphi_D = \varphi_{scf} \wedge s^{\mathbf{t}}_k$. As above, we can show that there is a conflict-free pair (X,Y) with $s_k \in X$ if and only if φ_D is satisfiable.*

NP-hard: *We provide a reduction from SAT. Let ψ be a propositional formula over vocabulary P. Define an ADF $D = \mathsf{RED}_1(P, \psi)$ as in Reduction 4.1. Due to Lemma 4.4 we know that there is a conflict-free pair (X,Y) with $z \in X$ if and only if ψ is satisfiable.* □

Again, Lemma 4.13 yields the same complexity for the naive semantics.

Corollary 4.16. *Consider any $\mathcal{I} \in \{\mathcal{G}, \mathcal{U}\}$.* $\mathsf{Cred}^{\mathcal{I}}_{nai}$ *is* NP*-complete.*

To verify that a given pair is naive, we have some more work to do. Recalling that a pair is naive iff it is conflict-free and $\leq_i$-maximal with respect to being conflict-free, we can see that to verify naivety we have to verify conflict-freeness (in NP) and verify that there is no properly $\leq_i$-greater conflict-free pair (in coNP).

Proposition 4.17. *Let $\mathcal{O} \in \{\mathcal{G}, \mathcal{U}\}$.* $\mathsf{Ver}^{\mathcal{O}}_{nai}$ *is* DP*-complete.*

Proof. Containment follows from Theorem 4.8, so it suffices to show DP*-hardness. We use a reduction from SAT-UNSAT. Let (ϕ, ψ) be a tuple of propositional formulas over vocabularies P_1 and P_2, respectively, with $P_1 \cap P_2 = \emptyset$. Construct an ADF D as follows:*

$$\begin{aligned} S &= P_1 \cup P_2 \cup \{x, y, z\} \\ \varphi_p &= \neg p \text{ for } p \in P_1 \cup P_2 \\ \varphi_x &= \phi \wedge \neg z \\ \varphi_y &= \psi \vee z \\ \varphi_z &= z \end{aligned}$$

Furthermore, define the pair $\bar{n} = (\{x,y\}, S)$. We now show that $\bar{n}$ is naive for $\mathcal{O}$ iff ϕ is satisfiable and ψ is unsatisfiable. (Notice that the proof only uses $\mathcal{O}''$ and thus works for $\mathcal{O} = \mathcal{G}$ and $\mathcal{O} = \mathcal{U}$.)

if: *Let ϕ be satisfiable and ψ be unsatisfiable. We show that $\bar{n}$ is naive for $\mathcal{O}$. We first show that $\bar{n}$ is conflict-free for $\mathcal{O}$:*

- *$\varphi_x^{\bar{n}} = \phi \wedge \neg z$ is satisfiable by presumption whence $s \in \mathcal{O}''(\bar{n})$;*
- *$\varphi_y^{\bar{n}} = \psi \vee z$ is satisfiable whence $s \in \mathcal{O}''(\bar{n})$.*

It remains to show that all $\bar{o}$ with $\bar{n} <_i \bar{o}$ are not conflict-free. Clearly, setting a $p \in P_1 \cup P_2$ to true or false in $\bar{n}$ violates conflict-freeness. Furthermore, x and y are already set, so the only two choices are setting z to true or false. If we set z to true, that is, consider the pair $(\{x, y, z\}, S)$, then we observe that the formula $\varphi_x^{(\{x,y,z\},S)} = \phi \wedge \neg\top$ is unsatisfiable whence $x \notin \mathcal{O}''(\{x, y, z\}, S)$ and the pair is not conflict-free. So the only remaining candidate is setting z to false, that is, the pair $\bar{o} = (\{x, y\}, S \setminus \{z\})$. Since ψ is unsatisfiable and does not mention x, y, z, also $\varphi_y^{\bar{o}} = \psi \vee \bot$ is unsatisfiable. Thus $y \notin \mathcal{O}''(\bar{o})$ and $\bar{o}$ is not conflict-free. It follows that $\bar{n}$ is naive for $\mathcal{O}$.

only if: *Let $\bar{n}$ be naive for $\mathcal{O}$. Assume to the contrary that ϕ is unsatisfiable or ψ is satisfiable.*

1. *ϕ is unsatisfiable. Then $\varphi_x^{\bar{n}} = \phi \wedge \neg z$ is unsatisfiable and $x \notin \mathcal{O}''(\bar{n})$. Thus $\bar{n}$ is not conflict-free, in contradiction to it being naive.*
2. *ψ is satisfiable. Define the pair $\bar{o} = (\{x, y\}, S \setminus \{z\})$. Since ψ does not mention x, y, z, also the formula $\varphi_y^{\bar{o}} = \psi \vee \bot$ is satisfiable. Then $\bar{o}$ is conflict-free with $\bar{n} <_i \bar{o}$, contradiction.*

Thus ϕ is satisfiable and ψ is unsatisfiable. □

For sceptical reasoning, we can do no better than verifying the absence of a naive pair where the statement in question is not true. The hardness proof proceeds by "laying a trap" for the conflict-free semantics: setting a statement s to true (for example) in a pair (X, Y) can be justified *locally* by the partially evaluated acceptance formula $\varphi_s^{(X,Y)}$ being satisfiable. However, the satisfying assignment for statements from $Y \setminus X$ need not pay respect to what might be dictated by other parts of the framework. The proof makes use of this and employs two new statements y and z with acceptance formulas $\varphi_y = z \rightarrow \psi$ for a given QBF matrix ψ and $\varphi_z = \top$. Now setting y to true can be locally justified since φ_y is satisfiable by setting z to false. However, in the case where ψ is unsatisfiable, we cannot set both y *and* z to true, since the respective partial evaluation of φ_y would be unsatisfiable, thereby violating conflict-freeness. Consequentially, this case leads to different kinds of naive pairs (those with $y \mapsto \mathbf{t}$ and $z \mapsto \mathbf{u}$, and those with $y \mapsto \mathbf{u}$ and $z \mapsto \mathbf{t}$). This reasoning is used in the proof below.

Proposition 4.18. *Let $\mathcal{O} \in \{\mathcal{G}, \mathcal{U}\}$. $\mathsf{Scep}_{nai}^{\mathcal{O}}$ is Π_2^P-complete.*

Proof. in Π_2^P: *Let $D = (S, L, C)$ be an ADF and $s \in S$. We can guess a pair (X, Y) with $s \notin X$ and verify in* DP *that it is naive.*

Π_2^P-hard: *We provide a reduction from the Π_2^P-complete problem of deciding whether a $QBF_{2,\forall}$-formula is valid. Let $\forall P \exists Q \psi$ be an instance of $QBF_{2,\forall}$-TRUTH where $P, Q \neq \emptyset$ and (w.l.o.g.) ψ mentions at least one $p \in P$ and at least one $q \in Q$. We construct an ADF D_ψ with a special statement z that is true in each naive pair of D_ψ if and only if $\forall P \exists Q \psi$ is true. Define $D_\psi = (S, L, C)$ with*

$$
\begin{aligned}
S &= P \cup Q \cup \{y, z\} \text{ with } y, z \notin P \cup Q \\
\varphi_p &= p \text{ for } p \in P \\
\varphi_q &= \neg q \text{ for } q \in Q \\
\varphi_y &= z \rightarrow \psi \\
\varphi_z &= \top
\end{aligned}
$$

if: *Let $\forall P \exists Q \psi$ be true; then for each $M \subseteq P$, the formula $\psi^{(M,M\cup Q)}$ is satisfiable. In particular, for all $M \subseteq N \subseteq P$, the formula $\psi^{(M,N\cup Q)}$ is satisfiable. We show that for each pair (X,Y) that is conflict-free for D_ψ, we can set z to true without violating conflict-freeness. Since z cannot be set to false by definition, it follows that z is true in every naive pair of D_ψ.*

Let (X,Y) be conflict-free for D_ψ. By definition of φ_z we have $z \in \mathcal{O}'(X,Y) \subseteq Y$. Furthermore, all $q \in Q$ are undefined in (X,Y), that is, $Q \subseteq Y \setminus X$. We have to show that $(X \cup \{z\}, Y)$ is conflict-free for D_ψ. Since $z \in Y$, the pair is consistent.

- *Let $t \in X$. By the assumption that (X,Y) is conflict-free, $\varphi_t^{(X,Y)}$ is satisfiable. If $t \in P$, then $\varphi_t^{(X\cup\{z\},Y)} = \varphi_t^{(X,Y)}$ is satisfiable. The case $t \in Q$ is impossible. If $t = y$, then $\varphi_t^{(X\cup\{z\},Y)} = \varphi_y^{(X\cup\{z\},Y)} = \top \to \psi^{(X,Y)} \equiv \psi^{(X,Y)}$ is satisfiable by assumption. The case $t = z$ is trivial.*
- *Let $f \in S \setminus Y$. If $f \in P$, then $\varphi_f^{(X\cup\{z\},Y)} = \varphi_f^{(X,Y)}$ and $f \notin \mathcal{O}'(X \cup \{z\}, Y)$ since $f \notin \mathcal{O}'(X,Y)$. The cases $f \in Q$ and $f = z$ are impossible. If $f = y$, then $y \notin \mathcal{O}'(X,Y)$.*
  1. *For $\mathcal{O} = \mathcal{U}_{D_\psi}$, this means that $\varphi_y^{(X,Y)} = (z \to \psi)^{(X,Y)}$ is refutable. Then the formula $\psi^{(X,Y)} = \psi^{(X\cup\{z\},Y)} \equiv \top \to \psi^{(X\cup\{z\},Y)} = \varphi_y^{(X\cup\{z\},Y)}$ is refutable and thus $y \notin \mathcal{U}'_{D_\psi}(X \cup \{z\}, Y)$.*
  2. *For $\mathcal{O} = \mathcal{G}_{D_\psi}$, the formula $\varphi_y^{(X\cup\{z\},Y)} = (z \to \psi)^{(X\cup\{z\},Y)}$ contains variables (ψ contains at least one $q \in Q \subseteq Y \setminus X$ by presumption) whence $y \notin \mathcal{G}'_{D_\psi}(X \cup \{z\}, Y)$.*

only if: *Let $M \subseteq P$ be such that the formula $\psi^{(M,M\cup Q)}$ is unsatisfiable. We show that the pair $(M \cup \{y\}, M \cup Q \cup \{y,z\})$ with $z \notin M$ is naive. Clearly, the pair is conflict-free for statements among $P \cup Q \cup \{z\}$. The formula $\varphi_y^{(M\cup\{y\},M\cup Q\cup\{y,z\})} = z \to \psi^{(M,M\cup Q)}$ is satisfiable (set z to false) whence $y \in \mathcal{O}''(M \cup \{y\}, M \cup Q \cup \{y,z\})$. The $p \in P$ are all set to true or false, and none of the $q \in Q$ can be set to true or false; our only possibility is to set z to true or false. Setting z to false is obviously not conflict-free by definition. For the pair $(M \cup \{y,z\}, M \cup Q \cup \{y,z\})$, we have that*

$$\varphi_y^{(M\cup\{y,z\},M\cup Q\cup\{y,z\})} = \top \to \psi^{(M\cup\{y,z\},M\cup Q\cup\{y,z\})} \equiv \psi^{(M,M\cup Q)}$$

is unsatisfiable whence $z \notin \mathcal{O}''(M \cup \{y,z\}, M \cup Q \cup \{y,z\})$ and the pair is not conflict-free. Therefore, $(M \cup \{y\}, M \cup Q \cup \{y,z\})$ is a naive pair where z is not true. □

Recalling that deciding existence of non-trivial ultimate conflict-free pairs is NP-hard, we can show that deciding existence of non-trivial *approximate* conflict-free pairs is (potentially) easier. The reason lies in the smaller precision of the approximate operator, as witnessed in Example 4.3.

Proposition 4.19. $\mathsf{Exists}^{\mathcal{G}}_{scf}$ *is in* P.

Proof. *Let $D = (S, L, C)$ be an ADF. If $S = \emptyset$ then there is no non-trivial pair at all, so assume $S \neq \emptyset$.*

1. *There is an $s \in S$ such that the formula $\varphi_s^{(\emptyset,S\setminus\{s\})}$ contains variables. Then by definition of $\mathcal{G}_D$, we have $s \notin \mathcal{G}'_D(\emptyset, S \setminus \{s\})$ and thus $(\emptyset, S \setminus \{s\})$ is conflict-free and non-trivial.*

2. *For all $s \in S$, the formula $\varphi_s^{(\emptyset, S\setminus\{s\})}$ is an expression consisting only of connectives and truth values. Then for each $s \in S$ the expression $\varphi_s^{(\emptyset, S\setminus\{s\})}$ has a fixed truth value, which can be computed in polynomial time.*

   (a) *There is an $s \in S$ such that $\varphi_s^{(\emptyset, S\setminus\{s\})} \equiv \bot$. Then by definition of $\mathcal{G}_D$, we have $s \notin \mathcal{G}'_D(\emptyset, S \setminus \{s\})$ and thus $(\emptyset, S \setminus \{s\})$ is conflict-free and non-trivial.*

   (b) *For all $s \in S$, we find $\varphi_s^{(\emptyset, S\setminus\{s\})} \equiv \top$. Then we consider the truth value of $\varphi_s^{(\{s\}, S)}$ for all $s \in S$, which is computable in polynomial time.*

      i. *There is an $s \in S$ for which we find $\varphi_s^{(\{s\}, S)} \equiv \top$. In particular, $\varphi_s^{(\{s\}, S)}$ is satisfiable, whence $s \in \mathcal{G}''_D(\{s\}, S)$. Thus the pair $(\{s\}, S)$ is conflict-free and non-trivial.*

      ii. *For all $s \in S$, we find $\varphi_s^{(\{s\}, S)} \equiv \bot$. Then for each $s \in S$, we have $\varphi_s \equiv \neg s$ and $(\emptyset, S)$ is the only conflict-free pair.* □

As usual, Lemma 4.13 yields the same bounds for the existence of non-trivial approximate naive pairs.

Corollary 4.20. $\mathsf{Exists}_{nai}^{\mathcal{G}}$ *is in* P.

However, these two results are the only ones where the complexities of the approximate and ultimate operators differ for semantics based on conflict-freeness.

4.2.2 Admissibility-based Semantics

We begin our complexity analysis of admissibility-based semantics by introducing and recalling some basic concepts of these semantics and the corresponding operators. By definition, a pair (X, Y) with $X \subseteq Y$ is admissible for operator $\mathcal{O}$ iff $(X, Y) \leq_i \mathcal{O}(X, Y)$. Since the operators under consideration are $\leq_i$-monotone, we can directly infer that if (X, Y) is admissible then it holds that $\mathcal{O}(X, Y) \leq_i \mathcal{O}(\mathcal{O}(X, Y))$. This means that (i) applying operator $\mathcal{O}$ to an admissible pair yields again an admissible pair, and (ii) iterative applications of $\mathcal{O}$ to an admissible pair (X, Y) always yield pairs (X', Y') such that $(X, Y) \leq_i (X', Y')$. In more detail, (ii) implies that a statement assigned to true or false in (X, Y) will keep this assignment if the operator is applied, while undecided statements may change their assignment. If (X, Y) is a fixpoint of $\mathcal{O}$ (i.e. a complete pair of $\mathcal{O}$), then the application of $\mathcal{O}$ also does not change the undecided statements.

Most complexity results in the following mainly rely on the operator for the upper bound $\mathcal{U}'' = \mathcal{G}''$, which is the same for both approximate and ultimate operators (Lemma 4.1). We also make use of the reductions from Section 4.1.3.

The first problem we analyse is the verification problem for admissible semantics. As before for most problems we need only show hardness since Theorem 4.8 shows membership.

Proposition 4.21. *Consider any $\mathcal{I} \in \{\mathcal{G}, \mathcal{U}\}$.* $\mathsf{Ver}_{adm}^{\mathcal{I}}$ *is* coNP*-complete.*

Proof. We provide a reduction from the problem of deciding whether a given formula ψ over vocabulary P is unsatisfiable. Let ADF $D = \mathsf{RED}_1(P, \psi)$ as defined in Reduction 4.1 and $\mathcal{O} \in \{\mathcal{G}_D, \mathcal{U}_D\}$. The pair $(\emptyset, P)$ is admissible for $\mathcal{O}$ iff $z \notin \mathcal{U}''_D(\emptyset, P)$ iff ψ is unsatisfiable due to Lemma 4.4. □

The result for $\mathsf{Ver}_{adm}^{\mathcal{U}}$ was also independently obtained by Wallner (2014, Proposition 4.1.9); we include it here because the reduction works for both operators.

For verifying if a given pair is complete the complexity increases to DP compared to just checking admissibility. Briefly put, the coNP part decides whether the given pair is a postfixpoint and the additional NP check is used to decide whether the pair is a prefixpoint. Together they decide whether the pair is a fixpoint, that is, constitutes a complete interpretation.

Proposition 4.22. *Consider any $\mathcal{I} \in \{\mathcal{G}, \mathcal{U}\}$.* $\mathsf{Ver}^{\mathcal{I}}_{com}$ *is* DP*-complete.*

Proof. For $\mathcal{I} = \mathcal{U}$ this result also appears in the work of Brewka et al. (2013, Corollary 7). However, the following reduction works for both operators.

We provide a reduction from the DP*-complete problem of determining whether a given formula ϕ is satisfiable and a given formula ψ is unsatisfiable. Let ϕ and ψ be arbitrary formulas over the disjoint vocabularies P_1 and P_2 respectively. Let $P = P_1 \cup P_2$. Construct the following ADF $D = (P \cup \{y, z\}, L, C)$:*

$$\begin{aligned} \varphi_p &= p && \text{for } p \in P \\ \varphi_y &= \neg y \wedge \phi \\ \varphi_z &= \psi \end{aligned}$$

Let $\mathcal{O} \in \{\mathcal{G}_D, \mathcal{U}_D\}$. We now prove that $(\emptyset, P \cup \{y\}) = \mathcal{O}(\emptyset, P \cup \{y\})$ if and only if ϕ is satisfiable and ψ is unsatisfiable.

if: *Independent of ϕ and ψ we know due to Lemma 4.3 that for each $p \in P$, we have $p \notin \mathcal{O}'(\emptyset, P \cup \{y\})$ and $p \in \mathcal{O}''(\emptyset, P \cup \{y\})$. Now since ϕ is satisfiable, we get that the formula $\varphi_y^{(\emptyset, P \cup \{y\})}$ is satisfiable and thus $y \in \mathcal{O}''(\emptyset, P \cup \{y\})$. Furthermore, since $\psi = \varphi_z$ is unsatisfiable it follows immediately that $z \notin \mathcal{O}''(\emptyset, P \cup \{y\})$. Therefore $(\emptyset, P \cup \{y\})$ is complete for $\mathcal{O}$.*

only if: *For the other direction assume that ϕ is unsatisfiable or ψ is satisfiable. For the first case suppose ϕ is unsatisfiable. Then $y \notin \mathcal{U}''_D(\emptyset, P \cup \{y\})$ and $(\emptyset, P \cup \{y\})$ is not complete. For the second case suppose ψ is satisfiable. Then $z \in \mathcal{U}''_D(\emptyset, P \cup \{y\})$ and likewise $(\emptyset, P \cup \{y\})$ is not complete.* □

Grounded semantics Next, we analyse the complexity of verifying that a given pair is the approximate Kripke-Kleene semantics (the grounded pair) of an ADF D, that is, the least fixpoint of $\mathcal{G}_D$. It turns out that verifying if a given pair is grounded has the same complexity as verifying if the pair is complete. A similar result for the ultimate grounded semantics has been obtained by Wallner (2014, Theorem 4.1.4). In our proof for approximate semantics below, showing membership is the tricky part. Our approach is to reduce the steps of the operator computation into propositional logic (a reduction introduced earlier). In particular we construct two formulas, of which we check one for satisfiability and the other for unsatisfiability. This gives us an interesting, yet technical proof of membership.

Theorem 4.23. $\mathsf{Ver}^{\mathcal{G}}_{grd}$ *is* DP*-complete.*

Proof. Let D be an ADF and $X \subseteq Y \subseteq S$.

in DP: *We provide a reduction to SAT-UNSAT by extending the construction of Reduction 4.2. We*

additionally define the formulas

$$\phi_{\leq_i} = \bigwedge_{s_i \notin X} \neg t_i \wedge \bigwedge_{s_i \in Y} u_i \qquad (T,U) \leq_i (X,Y)$$

$$\phi_{\geq_i} = \bigwedge_{s_i \in X} t_i \wedge \bigwedge_{s_i \notin Y} \neg u_i \qquad (T,U) \geq_i (X,Y)$$

$$\phi_{=} = \phi_{\leq_i} \wedge \phi_{\geq_i} \qquad (T,U) = (X,Y)$$

$$\phi_{<_i} = \phi_{\leq_i} \wedge \neg\phi_{\geq_i} \qquad (T,U) <_i (X,Y)$$

$$\psi_1 = \phi_{\mathsf{cfp}} \wedge \phi_{=} \qquad \mathcal{G}_D(T,U) = (T,U) \text{ with } T \subseteq U \text{ and } (T,U) = (X,Y)$$

$$\psi_2 = \phi_{\mathsf{cfp}} \wedge \phi_{<_i} \qquad \mathcal{G}_D(T,U) = (T,U) \text{ with } T \subseteq U \text{ and } (T,U) <_i (X,Y)$$

Intuitively, $\phi_{=}$ will be used to force (X,Y) to be a fixpoint of $\mathcal{G}_D$, and $\phi_{<_i}$ will be used to stipulate the existence of a fixpoint with strictly less information than (X,Y).

We claim that (1) ψ_1 is satisfiable iff (X,Y) is a consistent fixpoint of $\mathcal{G}_D$, and (2) ψ_2 is satisfiable iff there is a fixpoint $(T,U) <_i (X,Y)$ of $\mathcal{G}_D$. From this it follows that (ψ_1, ψ_2) is a positive instance of SAT-UNSAT iff (X,Y) is the Kripke-Kleene semantics of D.

1. *ψ_1 is satisfiable iff (X,Y) is a consistent fixpoint of $\mathcal{G}_D$.*

   "if": *Let (X,Y) be a consistent fixpoint of $\mathcal{G}_D$. Set $(T,U) = (X,Y)$, then by Lemma 4.5 there is an interpretation $I \subseteq P$ with $I \models \phi_{\mathsf{cfp}}$. By definition, we also have $I \models \phi_{=}$, whence $I \models \psi_1$ and ψ_1 is satisfiable.*

   "only if": *Let $\psi_1 = \phi_{\mathsf{cfp}} \wedge \phi_{=}$ be satisfiable. Then in particular ϕ_{cfp} is satisfiable and by Lemma 4.5 there is an interpretation $I \subseteq P$ such that its associated pair (T,U) is a consistent fixpoint of $\mathcal{G}_D$. Since additionally $I \models \phi_{=}$, it follows that $(T,U) = (X,Y)$.*

2. *ψ_2 is satisfiable iff there is a fixpoint $(T,U) <_i (X,Y)$ of $\mathcal{G}_D$.*

   "if": *Let $(T,U) <_i (X,Y)$ with $T \subseteq U$ and $\mathcal{G}_D(T,U) = (T,U)$. By Lemma 4.5 we can define a two-valued interpretation $I \subseteq P$ such that $I \models \phi_{\mathsf{cfp}}$. It is straightforward to show that $(T,U) <_i (X,Y)$ implies $I \models \phi_{<_i}$.*

   "only if": *Let $I \subseteq P$ be an interpretation with $I \models \psi_2$. Since in particular $I \models \phi_{\mathsf{cfp}}$, Lemma 4.5 yields a consistent fixpoint (T,U) of $\mathcal{G}_D$. As above, we can show that $(T,U) <_i (X,Y)$.*

DP-hard: *This follows from the proof in Proposition 4.22: The complete pair to verify there coincides with the Kripke-Kleene semantics of the constructed ADF.* □

Admissible: Existence non-trivial We next ask whether there exists a *non-trivial* admissible pair, that is, if at least one statement has a truth value other than unknown. Clearly, we can guess a pair and perform the coNP-check to show that it is admissible. The next result shows that this is also the best we can do.

Theorem 4.24. $\mathsf{Exists}^{\mathcal{G}}_{adm}$ *is Σ^P_2-complete.*

Proof. in Σ^P_2: *For ADF D, we guess a pair (X,Y) and verify that $X \subseteq Y$ and $(\emptyset, S) <_i (X,Y)$ in polynomial time, and $(X,Y) \leq_i \mathcal{G}_D(X,Y)$ using the* NP *oracle (Lemma 4.7, Items 1, 3, and 5).*

Σ_2^P-hard: *We provide a reduction from the Σ_2^P-hard problem $QBF_{2,\exists}$-TRUTH. Let $\exists P \forall Q \psi$ be a QBF. We define an ADF $D = (S, L, C)$ as follows:*

$$
\begin{aligned}
S &= P \cup \text{-}P \cup Q \cup \{z\} \text{ where } \text{-}P = \{\text{-}p \mid p \in P\}, \\
\varphi_p &= \neg z \wedge \neg \text{-}p \text{ for } p \in P, \\
\varphi_{\text{-}p} &= \neg z \wedge \neg p \text{ for } \text{-}p \in \text{-}P, \\
\varphi_q &= \neg q \text{ for } q \in Q, \\
\varphi_z &= \neg z \wedge \neg \psi.
\end{aligned}
$$

We show that D has a non-trivial admissible pair iff $\exists P \forall Q \psi$ is true.

"if": *Let $M \subseteq P$ be such that $\psi^{(M, M \cup Q)}$ is a tautology. We define a consistent pair (X, Y) by setting*

$$
\begin{aligned}
X &= M \cup \{\text{-}p \mid p \in P \setminus M\} \\
Y &= X \cup Q
\end{aligned}
$$

(X, Y) is obviously non-trivial, since $z \notin Y$. Note that by definition $p \in X$ iff $p \in M$, and $p \notin Y$ iff $p \notin M$, whence $\psi^{(M, M \cup Q)} = \psi^{(X,Y)}$. It remains to show that (X, Y) is admissible for D, that is, $X \subseteq \mathcal{G}'_D(X, Y)$ and $\mathcal{G}'_D(Y, X) \subseteq Y$.

$X \subseteq \mathcal{G}'_D(X, Y)$:

- *Let $p \in X$ for $p \in P$. Then by definition $\text{-}p \notin Y$ and $\varphi_p^{(X,Y)} = \neg\bot \wedge \neg\bot \equiv \top$ whence $p \in \mathcal{G}'_D(X, Y)$.*
- *Let $\text{-}p \in X$ for $\text{-}p \in \text{-}P$. Symmetric.*

$\mathcal{G}'_D(Y, X) \subseteq Y$:

- *Let $p \notin Y$ for $p \in P$. Then by definition $\text{-}p \in X$ and $\varphi_p^{(X,Y)} = \neg\bot \wedge \neg\top \equiv \bot$ whence $p \notin \mathcal{G}'_D(Y, X)$.*
- *Let $\text{-}p \notin Y$ for $\text{-}p \in \text{-}P$. Symmetric.*
- *Finally, for $z \notin Y$, we have $\varphi_z^{(X,Y)} = \neg\bot \wedge \neg\psi^{(X,Y)} \equiv \neg\psi^{(X,Y)} = \neg\psi^{(M, M \cup Q)}$. Since $\psi^{(M, M \cup Q)}$ is a tautology by presumption, $\neg\psi^{(M, M \cup Q)} = \neg\psi^{(X,Y)}$ is unsatisfiable and $z \notin \mathcal{G}''_D(X, Y) = \mathcal{G}'_D(Y, X)$.*

"only if": *Let (X, Y) be a non-trivial admissible pair for D. We have to show that $\exists P \forall Q \psi$ is true. Define $M = X \cap P$, we show that $\psi^{(M, M \cup Q)}$ is a tautology. We first observe that $Q \subseteq Y \setminus X$ by their acceptance conditions and since (X, Y) is admissible (thus in particular conflict-free and Lemma 4.3 applies). We next show $z \notin Y$.*

By the presumption that (X, Y) is non-trivial, we get that (1) $X \neq \emptyset$ or (2) $Y \subsetneq S$.

1. *$X \neq \emptyset$.*
   (a) *$z \in X$. Then $\varphi_z^{(X,Y)} = \neg\top \wedge \psi^{(X,Y)} \equiv \bot$ and $z \notin \mathcal{G}'_D(X, Y)$, that is, (X, Y) is not admissible. Contradiction.*
   (b) *$p \in X$. Then by admissibility $p \in \mathcal{G}'_D(X, Y)$, that is, the formula $\varphi_p^{(X,Y)} = (\neg z \wedge \neg\text{-}p)^{(X,Y)}$ is variable-free and evaluates to true, whence $z \notin Y$ and $\text{-}p \notin Y$.*
   (c) *$\text{-}p \in X$. Symmetric.*
2. *$Y \subsetneq S$.*

(a) $z \in S \setminus Y$. *This is what we want to show.*
(b) $p \in S \setminus Y$. *By admissibility* $p \notin \mathcal{G}'_D(Y,X)$ *and the partially evaluated acceptance formula* $\varphi_p^{(X,Y)} = (\neg z \wedge \neg\text{-}p)^{(X,Y)}$ *must be unsatisfiable. Since* $z \notin X$, *we get* $\text{-}p \in X$. *By item 1c above, we get* $z \notin Y$.
(c) $\text{-}p \in S \setminus Y$. *Symmetric.*

Hence $z \notin Y$. *Since* (X,Y) *is admissible,* $z \notin \mathcal{G}'_D(Y,X)$. *Thus the partially evaluated acceptance formula* $\varphi_z^{(X,Y)} = \neg\bot \wedge \neg\psi^{(X,Y)}$ *is unsatisfiable, that is,* $\neg\psi^{(X,Y)}$ *is unsatisfiable and* $\psi^{(X,Y)} = \psi^{(M,M\cup Q)}$ *is a tautology.* □

Again, we use one of the general existence results of Section 3.6: Theorem 3.46 directly leads to the next result, which considerably simplifies the complexity analysis of deciding the existence of non-trivial pairs for admissibility-based semantics.

Lemma 4.25. *Let* $(L,\sqsubseteq)$ *be a complete lattice and* $\mathcal{O}$ *an approximating operator on the CPO* $(L^c,\leq_i)$. *The following are equivalent:*

1. $\mathcal{O}$ *has a non-trivial admissible pair.*
2. $\mathcal{O}$ *has a non-trivial preferred pair.*
3. $\mathcal{O}$ *has a non-trivial complete pair.*

Proof. "$(1) \Longrightarrow (2)$": *Let* $(\bot_L,\top_L) <_i (x,y) \leq_i \mathcal{O}(x,y)$. *By Theorem 3.46, there is a preferred pair* $(p,q) \in L^c$ *for which* $(\bot_L,\top_L) <_i (x,y) \leq_i (p,q)$.

"$(2) \Longrightarrow (3)$": *By Theorem 3.10 (Strass, 2013, Theorem 3.10), every preferred pair is complete.*

"$(3) \Longrightarrow (1)$": *Any complete pair is admissible (Definition 2.2).* □

This directly shows the equivalence of the respective decision problems, that is, it holds that $\mathsf{Exists}^{\mathcal{A}}_{adm} = \mathsf{Exists}^{\mathcal{A}}_{pre} = \mathsf{Exists}^{\mathcal{A}}_{com}$. Recall that $\mathcal{A}$ contains all approximation operators defined on some consistent CPO of S-subset pairs for some set S. Regarding decision problems for querying, sceptical reasoning with respect to admissibility is trivial, since $(\emptyset,S)$ is always an admissible pair in any ADF and thus the answer to $\mathsf{Scep}^{\mathcal{A}}_{adm}$ is always "no". Furthermore, credulous reasoning with respect to admissible, complete and preferred semantics coincides.

Lemma 4.26. *Let* $\mathcal{I} \in \{\mathcal{G},\mathcal{U}\}$. *It holds that* $\mathsf{Cred}^{\mathcal{I}}_{adm} = \mathsf{Cred}^{\mathcal{I}}_{com} = \mathsf{Cred}^{\mathcal{I}}_{pre}$.

Proof. Let D *be an ADF,* $\mathcal{O} \in \{\mathcal{G}_D,\mathcal{U}_D\}$ *and* $s \in S$. *Assume* (X,Y) *with* $s \in X$ *is admissible for* $\mathcal{O}$, *then there exists a* (X',Y') *with* $(X,Y) \leq_i (X',Y')$ *which is preferred for* $\mathcal{O}$ *and where* $s \in X'$ *by Theorem 3.46. Since any preferred pair is also complete and any complete pair is also admissible the claim follows.* □

Lemma 4.25 implies the same complexity for the existence of non-trivial complete and preferred pairs.

Corollary 4.27. *Consider any* $\mathcal{I} \in \{\mathcal{G},\mathcal{U}\}$ *and* $\sigma \in \{com,pre\}$. $\mathsf{Exists}^{\mathcal{O}}_{\sigma}$ *is* Σ^P_2*-complete.*

By corollary to Theorem 4.23, the existence of a non-trivial approximate grounded pair can be decided in DP by testing whether the trivial pair $(\emptyset,S)$ is (not) a fixpoint of the approximate operator. The following result shows that this bound can be improved, even for the ultimate operator.

Proposition 4.28. *Consider any $\mathcal{I} \in \{\mathcal{G}, \mathcal{U}\}$. $\mathsf{Exists}^{\mathcal{I}}_{grd}$ is* coNP*-complete.*

Proof. Let D be an ADF. Obviously, D has a non-trivial approximate grounded semantics iff the trivial pair $(\emptyset, S)$ is not a fixpoint of $\mathcal{G}_D$, so we show that the co-problem (deciding whether $\mathcal{G}_D(\emptyset, S) = (\emptyset, S)$) is NP*-complete.*

in NP: *We have that $\mathcal{G}_D(\emptyset, S) = (\emptyset, S)$ iff $\emptyset \subseteq \mathcal{G}'_D(\emptyset, S) \subseteq \emptyset$ and $S \subseteq \mathcal{G}''_D(\emptyset, S) \subseteq S$. So mainly we have to verify $\mathcal{G}'_D(\emptyset, S) \subseteq \emptyset$ and $S \subseteq \mathcal{G}''_D(\emptyset, S)$. By Lemma 4.7, the first part can be decided in* P *(item 1) and the second part in* NP *(item 4).*

NP-hard: *We give a reduction from SAT. Let ψ be a propositional formula over vocabulary P. Define an ADF $D = (S, L, C)$ with $S = P \cup \{z\}$ for $z \notin P$ and $\varphi_p = p$ for $p \in P$ and $\varphi_z = z \wedge \psi$. It is readily verified that by definition every statement has a parent that is undecided in $(\emptyset, S)$ and thus $\mathcal{G}'_D(\emptyset, S) = \emptyset$. Furthermore, $P \subseteq \mathcal{G}'_D(S, \emptyset)$ is easy to show. Thus $S \subseteq \mathcal{G}'_D(S, \emptyset)$ iff $z \in \mathcal{G}'_D(S, \emptyset)$ iff there is an $M \subseteq S$ with $par(z) \setminus M \subseteq S \setminus \emptyset$ and $M \models \varphi_z$ iff there is an $M \subseteq S$ with $M \models \varphi_z$ iff $\varphi_z = z \wedge \psi$ is satisfiable iff ψ is satisfiable.*

For $\mathcal{I} = \mathcal{U}$, the proof is analogous to the one above – we show NP*-completeness of the complementary problem.*

in NP: *We have to verify $\mathcal{U}'_D(\emptyset, S) \subseteq \emptyset$ and $S \subseteq \mathcal{U}''_D(\emptyset, S)$. By Lemma 4.7(2) and Lemma 4.7(4), this can be done in* NP*.*

NP-hard: *The construction is the same as for $\mathcal{I} = \mathcal{G}$.* □

Using the result for existence of non-trivial admissible pairs, the verification complexity for the preferred semantics is straightforward to obtain, similarly as in the case of AFs (Dimopoulos and Torres, 1996). The result for the ultimate semantics is due to Wallner (2014, Proposition 4.1.14).

Proposition 4.29. *Consider any $\mathcal{I} \in \{\mathcal{G}, \mathcal{U}\}$. $\mathsf{Ver}^{\mathcal{I}}_{pre}$ is Π^p_2-complete.*

Proof. in Π^p_2: *Let D be an ADF and $X \subseteq Y \subseteq S$. To show that (X, Y) is not preferred, we guess a pair (M, N) with $(X, Y) <_i (M, N)$ and use the* NP *oracle to show that (M, N) is a complete pair (which can be done in* DP*).*

Π^p_2-hard: *Consider the complementary problem of deciding whether a given pair is not a preferred pair. Even for the special case of the pair $(\emptyset, S)$, Theorem 4.24 shows that this problem is Σ^P_2-hard.* □

Entailment problems We now move on to query-based reasoning. Similarly as before, we mainly utilise the operator for the upper bound to show hardness. Due to this reason, and for the sake of uniformity of proving results for both operators, we slightly deviate from our definition of credulous and sceptical reasoning and show hardness for the question whether a σ-pair exists such that the given statement is false, respectively ask whether the statement is false in all σ-pairs. For the admissibility-based semantics it is straightforward to see that these problems can be reduced to each other. For querying statement s in an ADF D consider the modified ADF $\hat{D}$ with a fresh statement $\hat{s}$ and acceptance condition $\varphi_{\hat{s}} = \neg s$. Let $\mathcal{O} \in \{\mathcal{G}_D, \mathcal{U}_D\}$ and $\hat{\mathcal{O}} \in \{\mathcal{G}_{\hat{D}}, \mathcal{U}_{\hat{D}}\}$. It holds that (X, Y) is admissible for $\mathcal{O}$ with $s \in (S \setminus Y)$ iff $(X \cup \{\hat{s}\}, Y)$ is admissible for $\hat{\mathcal{O}}$. Likewise, it holds that for all preferred pairs (X, Y) of $\mathcal{O}$ we have $s \in (S \setminus Y)$ iff in all preferred pairs $(\hat{X}, \hat{Y})$ of $\hat{\mathcal{O}}$ we have $\hat{s} \in \hat{X}$. Further, s is false in the grounded pair of $\mathcal{O}$ iff $\hat{s}$ is true in the grounded pair of $\hat{\mathcal{O}}$.

We now show that on general ADFs credulous reasoning with respect to approximate admissibility is harder than on AFs. By Lemma 4.26, the same lower bound holds for complete and preferred semantics. The proof is obtained by the same reduction as the one for the ultimate admissible semantics (Wallner, 2014, Proposition 4.1.11).

Proposition 4.30. $\mathsf{Cred}^{\mathcal{G}}_{adm}$ *is* Σ^P_2*-complete.*

Proof. Membership is given by Theorem 4.8. Hardness is shown by a reduction from the Σ^P_2*-hard problem* $QBF_{2,\exists}$*-TRUTH.*

Let $\exists P \forall Q \psi$ *be a QBF. We define an ADF* D *as follows:*

$$S = P \cup Q \cup \{f\}$$
$$\varphi_p = p \text{ for } p \in P$$
$$\varphi_q = \neg q \text{ for } q \in Q$$
$$\varphi_f = \neg f \wedge \neg \psi$$

We now show that there exists an admissible pair (X, Y) *for* $\mathcal{G}_D$ *with* $f \in S \setminus Y$ *iff* $\exists P \forall Q \psi$ *is true. Note that for any admissible pair we have that* f *is not set to true, since then its acceptance condition evaluates to false.*

"if": *Assume the QBF is valid. Then there exists a* $P' \subseteq P$ *such that for any* $Q' \subseteq Q$ *we have* $P' \cup Q' \models \psi$*. We now show that* $(P', P' \cup Q)$ *is admissible for* $\mathcal{G}_D$*. Since for any* $p \in P'$ *we have that* $\{p\} \models \varphi_p$ *it follows that* $P' \subseteq \mathcal{G}'_D(P', P' \cup Q)$ *and* $P' \subseteq \mathcal{U}'_D(P', P' \cup Q)$*. Further* $\emptyset \models \varphi_q$ *for* $q \in Q$*, thus* $Q \subseteq \mathcal{G}''_D(P', P' \cup Q)$*. Lastly,* $R \not\models \varphi_f$ *for* $P' \subseteq R \subseteq P' \cup Q$ *since* $R \models \psi$*, hence* $f \notin \mathcal{G}''_D(P', P' \cup Q)$*.*

"only if": *Assume that there exists an admissible pair* (X, Y) *for* $\mathcal{G}_D$ *such that* $f \in S \setminus Y$*. First we show that* $Q \subseteq (Y \setminus X)$*. Suppose the contrary, then* (X, Y) *would not be admissible* $\mathcal{G}_D$*, since if there is a* $q \in Q$ *and* $q \in X$*, then* $q \notin \mathcal{G}''_D(X, Y)$*. Similarly* $q \in S \setminus Y$ *implies that* q *is in the new lower bound. We now show that* $X \cup Q' \models \psi$ *for any* $Q' \subseteq Q$*, thus implying that* ψ *is valid (note that* $X \subseteq P$*). Suppose there exists a* $Q' \subseteq Q$ *such that* $X \cup Q' \not\models \psi$*, then* $X \cup Q' \models \varphi_f$ *and* $X \subseteq (X \cup Q') \subseteq Y$ *and thus* $f \in \mathcal{G}''_D(X, Y)$*, implying that* (X, Y) *is not admissible, which is a contradiction.* □

For credulous and sceptical reasoning with respect to the grounded semantics, we first observe that the two coincide since there is always a unique grounded pair. Furthermore, a statement s is true in the approximate grounded pair iff s is true in the least fixpoint (of $\mathcal{G}_D$) iff s is true in all fixpoints iff there is no fixpoint where s is undecided or false. This condition can be encoded in propositional logic and leads to the next result. For ultimate semantics and coNP-hardness the proof of Brewka and Woltran (2010, Proposition 13) can be easily adapted (Wallner, 2014, Proposition 4.1.3).

Proposition 4.31. *Consider any* $\mathcal{I} \in \{\mathcal{G}, \mathcal{U}\}$*. Both* $\mathsf{Cred}^{\mathcal{I}}_{grd}$ *and* $\mathsf{Scep}^{\mathcal{I}}_{grd}$ *are* coNP*-complete.*

Proof. We give the proof for $\mathcal{I} = \mathcal{G}$*, the case* $\mathcal{I} = \mathcal{U}$ *is treated by Wallner (2014, Proposition 4.1.3).*

in coNP: *We reduce to unsatisfiability checking in propositional logic. Let* $D = (S, L, C)$ *be an ADF with* $S = \{s_1, \ldots, s_n\}$ *and assume we want to verify that* s_k *is true in the grounded pair of* D *for some* $1 \leq k \leq n$*. We again extend Reduction 4.2; additionally define the formulas*

$$\phi'_{\mathsf{cfp}} = \phi_{\mathsf{cfp}}[p/p' : p \in P] \qquad \text{(renamed copy of } \phi_{\mathsf{cfp}}\text{)}$$
$$\psi = (\phi_{\mathsf{fp}} \wedge \neg t_k \wedge u_k) \vee (\phi'_{\mathsf{fp}} \wedge \neg u'_k)$$

We claim that ψ *is unsatisfiable iff there is no consistent fixpoint where* s_k *is unknown or false.*

1. $\phi_{\mathsf{cfp}} \wedge \neg t_k \wedge u_k$ *is unsatisfiable iff there is no consistent fixpoint where* s_k *is undefined:*
   "if": *Let* $\phi_{\mathsf{cfp}} \wedge \neg t_k \wedge u_k$ *be satisfiable. Then there is an interpretation* $I \subseteq P$ *such that* $I \models \phi_{\mathsf{cfp}}$ *and* $I \models \neg t_k \wedge u_k$*. Using Lemma 4.5 we can construct a consistent pair* (T, U) *and show that it is a fixpoint of* $\mathcal{G}_D$ *with* $s_k \in U \setminus T$*.*
   "only if": *Let* $T \subseteq U \subseteq S$ *such that* $\mathcal{G}_D(T, U) = (T, U)$ *and* $s_k \in U \setminus T$*. Lemma 4.5 yields an interpretation* $I \subseteq P$ *such that* $I \models \phi_{\mathsf{cfp}}$ *and* $I \models \neg t_k \wedge u_k$*.*
2. $\phi'_{\mathsf{cfp}} \wedge \neg u'_k$ *is unsatisfiable iff there is no consistent fixpoint where* s_k *is false: similar.*

coNP-hard: *Let* ψ *be a propositional formula over vocabulary* P*. Define the ADF* $D = (S, L, C)$ *with* $S = P \cup \{z\}$*,* $\varphi_p = \neg\psi$ *for* $p \in P$*, and* $\varphi_z = \bigwedge_{p \in P} \neg p$*. We show that* z *is true in the grounded semantics of* $\mathcal{G}_D$ *iff* ψ *is a tautology.*

"if": *Let* ψ *be a tautology. Then* $\neg\psi$ *is unsatisfiable and* $p \notin \mathcal{G}''_D(\emptyset, S)$ *for all* $p \in P$*. Obviously* φ_z *is satisfiable whence* $z \in \mathcal{G}''_D(\emptyset, S)$*. Thus* $\mathcal{G}''_D(\emptyset, S) = \{z\}$*. Furthermore* $\mathcal{U}'_D(\emptyset, S) = \emptyset$*, since no acceptance condition is a tautology. Therefore also* $\mathcal{G}'_D(\emptyset, S) = \emptyset$*. Thus* $\mathcal{G}_D(\emptyset, S) = (\emptyset, \{z\})$*. Now since* z *does not occur in the acceptance formula of* z*, it is clear that* $\mathcal{G}_D(\emptyset, \{z\}) = (\{z\}, \{z\}) = \mathcal{G}_D(\{z\}, \{z\})$*. Thus* z *is true in the grounded semantics of* $\mathcal{G}_D$*.*

"only if": *Let* $lfp(\mathcal{G}_D) = (X, Y)$ *and* $z \in X$*. By the acceptance condition of* z *and the fact that* (X, Y) *is a fixpoint of* $\mathcal{G}_D$ *we get* $P \cap Y = \emptyset$*. Since* $X \subseteq Y$ *we have* $(X, Y) = (\{z\}, \{z\})$*. Assume to the contrary that* ψ *is not a tautology. Then* $\neg\psi$ *is satisfiable and* $P \subseteq Y = \mathcal{G}''_D(\emptyset, S)$*. Contradiction.* □

Regarding sceptical reasoning for the remaining semantics we need only show the results for complete and preferred semantics, in all other cases the complexity coincides with credulous reasoning or is trivial. For complete semantics it is easy to see that sceptical reasoning coincides with sceptical reasoning under grounded semantics, since the grounded pair is the $\leq_i$-least complete pair.

Corollary 4.32. *Consider any* $\mathcal{I} \in \{\mathcal{G}, \mathcal{U}\}$*.* $\mathsf{Scep}^{\mathcal{I}}_{com}$ *is* coNP*-complete.*

Similar to reasoning on AFs, we step up one level of the polynomial hierarchy by changing from credulous to sceptical reasoning with respect to preferred semantics, which makes sceptical reasoning under preferred semantics particularly hard. The proof for the ultimate semantics uses proof ideas by Dunne and Bench-Capon (2002) and can be found in the work of Wallner (2014, Theorem 4.1.17). The approximate semantics can be treated in a similar way (Strass and Wallner, 2015, Theorem 4.22), and we will not reproduce the proof here in its entirety.

Theorem 4.33. *Consider any* $\mathcal{I} \in \{\mathcal{G}, \mathcal{U}\}$*.* $\mathsf{Scep}^{\mathcal{I}}_{pre}$ *is* Π^P_3*-complete.*

Proof. Membership is given by Theorem 4.8. Hardness is shown by a reduction from the Π^P_3*-hard problem* $QBF_{3,\forall}$*-TRUTH. Let* $\forall P \exists Q \forall R \psi$ *be a QBF. We define an ADF* D *as follows:*

$$
\begin{aligned}
S &= P \cup Q \cup \text{-}Q \cup R \cup \{f\} && \text{with } \text{-}Q = \{\text{-}q \mid q \in Q\} \\
\varphi_p &= p && \text{for } p \in P \\
\varphi_q &= \neg f \wedge \neg\text{-}q && \text{for } q \in Q \\
\varphi_{\text{-}q} &= \neg f \wedge \neg q && \text{for } \text{-}q \in \text{-}Q \\
\varphi_r &= \neg r && \text{for } r \in R \\
\varphi_f &= \neg f \wedge \neg\psi
\end{aligned}
$$

The proof proceeds by showing that all preferred pairs (X, Y) of $\mathcal{G}_D$ have $f \in S \setminus Y$ iff $\forall P \exists Q \forall R \psi$ is true. For the remaining technicalities, we refer the interested reader to the works of Wallner (2014, Theorem 4.1.17) and Strass and Wallner (2015, Theorem 4.22). □

4.2.3 Two-valued Semantics

The complexity results we have obtained so far might lead the reader to ask why we bother with the approximate operator $\mathcal{G}_D$ at all: the ultimate operator $\mathcal{U}_D$ is at least as precise and for all admissibility-based and most conflict-free-based semantics considered up to now, it has the same computational costs. We now show that for the verification of two-valued stable models, the operator for the upper bound plays no role and therefore the complexity difference between the lower bound operators for approximate (in P) and ultimate (coNP-hard) semantics comes to bear.

For the ultimate two-valued stable semantics, Brewka et al. (2013) already have some complexity results: model verification is in DP (Proposition 8), and model existence is Σ_2^P-complete (Theorem 9). We will show next that we can do better for the approximate version.

Proposition 4.34. *Let D be an ADF and $X \subseteq Y \subseteq S$. Checking that X is the least fixpoint of $\mathcal{G}'_D(\cdot, Y)$ can be done in polynomial time.*

Proof. We provide the following polynomial-time decision procedure with input D, X, Y.

1. *Set $i = 0$ and $X_0 = \emptyset$.*
2. *For each statement $s \in S$, do the following:*
   (a) *If $par(s) \cap (Y \setminus X_i) = \emptyset$ and $par(s) \cap X_i \models \varphi_s$, then set $s \in X_{i+1}$.*
3. *If $X_{i+1} = X_i = X$ then return "Yes".*
4. *If $X_{i+1} = X_i \subsetneq X$ then return "No".*
5. *If $X_{i+1} \not\subseteq X$ then return "No".*
6. *Increment i and go to step 2.*

Overall, the loop between steps 2 and 6 is executed at most $|S|$ times, since $X_i \subseteq X_{i+1}$ for all $i \in \mathbb{N}$ and we can add at most all statements one by one. In each execution of the loop, step 2a is executed $|S|$ times. The conditions of step 2a, in particular $par(s) \cap X_i \models \varphi_s$, can be verified in polynomial time.

It remains to show that X is the least fixpoint of $\mathcal{G}'_D(\cdot, Y)$ iff the procedure returns "Yes".

"if": *Assume the procedure returned "Yes" on input D, X, Y.*

- *X is a fixpoint of $\mathcal{G}'_D(\cdot, Y)$, that is, $\mathcal{G}'_D(X, Y) = X$:*
  "$\subseteq$": *Let $s \in \mathcal{G}'_D(X, Y)$. Then there is a $B \subseteq X \cap par(s)$ such that $C_s(B) = \mathbf{t}$ and $par(s) \setminus B \subseteq S \setminus Y$. As in the proof of Proposition 4.6, we get that $B = X \cap par(s)$, $C_s(par(s) \cap X) = \mathbf{t}$ and $par(s) \cap (Y \setminus X) = \emptyset$. Since the procedure answered "Yes", there was an $i \in \mathbb{N}$ with $X_{i+1} = X_i = X$. From step 2a of the procedure, we know that $par(s) \cap (Y \setminus X_i) = \emptyset$ and $C_s(par(s) \cap X_i) = \mathbf{t}$ means that $s \in X_{i+1} = X$.*
  "$\supseteq$": *Let $s \in X$. Since the procedure answered "Yes", there was an $i \in \mathbb{N}$ with $X_{i+1} = X_i = X$. Now $s \in X_{i+1}$ by step 2a of the procedure means that $par(s) \cap (Y \setminus X_i) = \emptyset$ and $C_s(par(s) \cap X_i) = \mathbf{t}$. Thus there exists a $B = par(s) \cap X$ with $C_s(B) = \mathbf{t}$ and $par(s) \setminus B \subseteq S \setminus Y$, and $s \in \mathcal{G}'_D(X, Y)$.*

- *X is the least fixpoint: Assume to the contrary that there is some $X' \subsetneq X$ that is a fixpoint of $\mathcal{G}'_D(\cdot, Y)$. But then step 4 of the procedure would have detected $X_{i+1} = X_i = X' \subsetneq X$ and returned "No", contradiction.*

"only if": *Let X be the least fixpoint of $\mathcal{G}'_D(\cdot, Y)$ and assume to the contrary that the procedure answered "No".*

- *The procedure answered "No" in step 4. By the argument above, we can show that there is a fixpoint $X' \subsetneq X$, contradiction.*
- *The procedure answered "No" in step 5. We have $X_{i+1} \not\subseteq X$ for some $i \in \mathbb{N}$, that is, there is some $s \in X_{i+1}$ with $s \notin X$. Since $s \in X_{i+1}$, we have $par(s) \cap (Y \setminus X_i) = \emptyset$ and $C_s(par(s) \cap X_i) = \mathbf{t}$. Since the procedure did not terminate with X_i already, we know that $X_i \subseteq X$. Therefore, $par(s) \cap (Y \setminus X) = \emptyset$ and $C_s(par(s) \cap X) = \mathbf{t}$. This means $s \in \mathcal{G}'_D(X, Y) = X$. Contradiction.* □

In particular, the procedure can decide whether Y is the least fixpoint of $\mathcal{G}'_D(\cdot, Y)$, that is, whether (Y, Y) is a two-valued stable model of $\mathcal{G}_D$. This yields the next result.

Theorem 4.35. *1. $\mathsf{Ver}^{\mathcal{G}}_{stm}$ is in P, and*

2. $\mathsf{Exists}^{\mathcal{G}}_{stm}$ is NP-complete.

Proof. Let D be an ADF and $X \subseteq S$.

1. *We have to verify that X is the least fixpoint of the operator $\mathcal{G}'_D(\cdot, X)$, which can be done in polynomial time by Proposition 4.34.*

2. *Deciding whether D has an approximate two-valued stable model is NP-complete:*

   in NP: *To decide whether D has an approximate two-valued stable model, we guess a set $X \subseteq S$ and verify as above that (X, X) is indeed a two-valued stable model of $\mathcal{G}_D$.*

   NP-hard: *Carries over from AFs.* □

The hardness direction of the second part is clear since the respective result from stable semantics of abstract argumentation frameworks carries over.

Brewka et al. (2013) showed that $\mathsf{Ver}^{\mathcal{U}}_{stm}$ is in DP (Proposition 8). We can improve that upper bound to coNP: basically the operator for the upper bound (contributing the NP part) is not really needed. We furthermore also provide a hardness proof for coNP.

Proposition 4.36. *$\mathsf{Ver}^{\mathcal{U}}_{stm}$ is coNP-complete.*

Proof. in coNP: *Given an ADF $D = (S, L, C)$ and a set $M \subseteq S$ we first construct the reduct D^M in polynomial time. Now M is an ultimate two-valued stable model of D iff all statements in M are true in the grounded semantics of D^M and (M, M) is a model of D. Verifying if a statement is true in the ultimate grounded pair of an ADF is coNP-complete due to Proposition 4.31. Thus verifying that all statements in M are true in the ultimate grounded pair is likewise a problem in coNP. Verifying if (M, M) is a model of D can be achieved in polynomial time. This means that $\mathsf{Ver}^{\mathcal{U}}_{stm}$ is in coNP.*

coNP-hard: *Let ψ be a propositional formula over a vocabulary P. We define an ADF D over statements P with $\varphi_p = \psi$ for all $p \in P$. When we apply $\mathcal{U}'_D$ to the pair $(\emptyset, P)$, there are only two possible outcomes: either $\psi = \psi^{(\emptyset,P)} = \varphi_p^{(\emptyset,P)}$ is a tautology, then $p \in \mathcal{U}'_D(\emptyset, P)$ for all $p \in P$, that is $\mathcal{U}'_D(\emptyset, P) = P$; otherwise ψ is refutable and accordingly $\mathcal{U}'_D(\emptyset, P) = \emptyset$. Furthermore, in the former case it follows from $\leq_i$-monotonicity of $\mathcal{U}_D$ that $P = \mathcal{U}'_D(\emptyset, P) \subseteq \mathcal{U}'_D(P, P)$. Thus ψ is a tautology if and only if P is a fixpoint of $\mathcal{U}'_D(\cdot, P)$ and $\emptyset$ is not. Now*

$$
\begin{aligned}
&\textit{P is an ultimate two-valued stable model of D}\\
&\textit{iff P is the least fixpoint of } \mathcal{U}'_D(\cdot, P)\\
&\textit{iff } \mathcal{U}'_D(\emptyset, P) = P = \mathcal{U}'_D(P, P)\\
&\textit{iff ψ is a tautology}
\end{aligned}
$$

□

We now turn to the credulous and sceptical reasoning problems for the two-valued semantics. We first recall that a two-valued pair (X, X) is a supported model (or model for short) of an ADF D iff $\mathcal{G}_D(X, X) = (X, X)$. Thus it could equally well be characterised by the two-valued operator by saying that X is a model iff $G_D(X) = X$. Now since $\mathcal{U}_D$ is the ultimate approximation of G_D, also $\mathcal{U}_D(X, X) = (X, X)$ in this case. Rounding up, this recalls that approximate and ultimate two-valued supported models coincide. Hence we get the following results for reasoning with this semantics.

Corollary 4.37. *Consider any $\mathcal{I} \in \{\mathcal{G}, \mathcal{U}\}$. $\mathsf{Cred}^{\mathcal{I}}_{mod}$ is NP-complete and $\mathsf{Scep}^{\mathcal{I}}_{mod}$ is coNP-complete.*

Proof. The membership parts are clear since $\mathsf{Ver}^{\mathcal{I}}_{mod}$ is in P. Hardness carries over from AFs (Dimopoulos and Torres, 1996). □

For the approximate two-valued stable semantics, the fact that model verification can be decided in polynomial time leads to the next result.

Corollary 4.38. *$\mathsf{Cred}^{\mathcal{G}}_{stm}$ is NP-complete and $\mathsf{Scep}^{\mathcal{G}}_{stm}$ is coNP-complete.*

Proof. The membership parts are clear since $\mathsf{Ver}^{\mathcal{G}}_{stm}$ is in P. Hardness carries over from AFs (Dimopoulos and Torres, 1996). □

For the ultimate two-valued stable semantics, things are bit more complex. The following result was already presented by Brewka et al. (2013), however they had to leave out the proof due to space restrictions. We present the proof (following the proof of Theorem 6.12 by Denecker et al., 2004) here for completeness and since we will need it later on.

Theorem 4.39. *$\mathsf{Exists}^{\mathcal{U}}_{stm}$ is Σ_2^{P}-complete.*

Proof. Let $D = (S, L, C)$ be an ADF. For membership, we first guess a set $M \subseteq S$. We can verify in polynomial time that M is a two-valued supported model of D, and compute the reduct D_M. Using the NP oracle, we can compute the grounded semantics (K', K'') of the reduct in polynomial time. It then only remains to check $K' = M$.

For hardness, we provide a reduction from the Σ_2^P-complete problem of deciding whether a $QBF_{2,\exists}$-formula is valid. Let $\exists P \forall Q \psi$ be an instance of $QBF_{2,\exists}$-TRUTH where ψ is in DNF and $P, Q \neq \emptyset$. We have to construct an ADF D such that D has a stable model iff $\exists P \forall Q \psi$ is true.

First of all, define $\text{-}P = \{\text{-}p \mid p \in P\}$ for abbreviating the negations of $p \in P$. For guessing an interpretation for P, define the acceptance formulas $\varphi_p = \neg\text{-}p$ and $\varphi_{\text{-}p} = \neg p$ for $p \in P$. Define ψ' as the formula $\psi[\neg p/\text{-}p]$ where all occurrences of $\neg p$ have been replaced by $\text{-}p$. Further add a statement

z with $\varphi_z = \neg z \wedge \neg\psi'$, an integrity constraint that ensures truth of ψ' in any model. For $q \in Q$ we set $\varphi_q = \psi'$. Thus we get the statements $S = P \cup \text{-}P \cup Q \cup \{z\}$. We have to show that D has a stable model iff $\exists P \forall Q \psi$ is true.

"if": *Let $M_P \subseteq P$ be such that the following formula over vocabulary Q is a tautology:*

$$\phi = \psi^{(M_P, M_P \cup Q)}$$

We now construct a stable model $M = M_P \cup Q \cup \{\text{-}p \in \text{-}P \mid p \notin M_P\}$. We first show that M is a model of D: For each $p \in M_P$, we have $\text{-}p \notin M$ by definition and hence $M \models \varphi_p = \neg\text{-}p$. Conversely, if $p \notin M_P$ then $\text{-}p \in M$ and $M \models \varphi_{\text{-}p} = \neg p$. For $q \in Q$, we have that $\varphi_q = \psi'$ and so we have to show $M \models \psi'$. This is however immediate since ϕ (the partial evaluation of ψ with M as interpretation for P) is a tautology. Finally, by definition $z \notin M$, and since $M \models \psi'$ we get $M \not\models \varphi_z = \neg z \wedge \neg\psi'$ as required.

To show that M is a stable *model, we have to show that all statements in M are true in the ultimate Kripke-Kleene semantics of the reduct D_M. The reduct is given by $D_M = (M, L_M, C_M)$ with*

$$\begin{aligned} \varphi_p &= \neg\bot && \text{for } p \in M \\ \varphi_{\text{-}p} &= \neg\bot && \text{for } \text{-}p \in M \\ \varphi_q &= \psi'^{(\emptyset, M)} \end{aligned}$$

The computation of the Kripke-Kleene semantics starts with $(\emptyset, M)$ and leads to the first revision $(K_0', K_0'') = \mathcal{U}_D(\emptyset, M)$. Since the acceptance condition of any $p, \text{-}p \in M$ is tautological, we have $p, \text{-}p \in K_0'$, that is, the statements $p, \text{-}p \in M$ are considered true. For the next step, the acceptance formula of any $q \in Q$ can thus be simplified to

$$\begin{aligned} \varphi_q^{(M \setminus Q, M)} &= \left(\psi'^{(\emptyset, M)}\right)^{(M \setminus Q, M)} \\ &= \psi'^{(M \setminus Q, M)} \\ &= \psi'[p/\bot : p \notin M, \text{-}p/\bot : \text{-}p \notin M, p/\top : p \in M, \text{-}p/\top : \text{-}p \in M], \end{aligned}$$

a formula over Q that is equivalent to $\phi = \psi^{(M_P, M_P \cup Q)}$. By presumption, ϕ is a tautology. Hence at this point all acceptance formulas partially evaluated by (K_0', K_0'') are tautologies and thus $\mathcal{U}_D(K_0', K_0'') = (M, M)$, which has already been shown to be a fixpoint of $\mathcal{U}_D$.

"only if": *Let $M \subseteq S$ be an ultimate two-valued stable model of D. We have to show that $\exists P \forall Q \psi$ is true. Define $M_P = M \cap P$ and $\phi = \psi^{(M_P, M_P \cup Q)}$. We show that ϕ is a tautology.*

First of all, since M is a model of D_M we have $z \notin M$: assume to the contrary that $z \in M$, then M is a model for $\varphi_z = \neg z \wedge \neg\psi' \equiv \bot \wedge \neg\psi'$, contradiction. Hence $M \not\models \neg z \wedge \neg\psi'$, that is, $M \not\models \neg\psi'$. This shows that $M \models \psi'$, that is, $M \models \varphi_q$ for all $q \in Q$, whence $Q \subseteq M$. Thus the evaluation of $p \in P$ and $\text{-}p \in \text{-}P$ defined by M shows the truth of the formula

$$\psi'^{(M,M)} = \psi'[p/\top : p \in M, \text{-}p/\top : \text{-}p \in M, p/\bot : p \notin M, \text{-}p/\bot : \text{-}p \notin M][q/\top : q \in Q]$$

Now since M is a stable model of D, the pair (M, M) is the ultimate grounded semantics of the reduct D_M as above. To show that ϕ is a tautology, assume to the contrary that ϕ is refutable. As observed in the "if" part, ϕ is equivalent to the formula $\varphi_q^{(M \setminus Q, M)}$. Thus also φ_q is refutable, whence $q \notin \mathcal{U}'_{D_M}(\emptyset, M)$ for all $q \in Q$ and $\mathcal{U}'_{D_M}(\emptyset, M) = M \setminus Q$. Furthermore we know that $\mathcal{U}''_{D_M}(\emptyset, M) = M$. Now $\varphi_q^{(M \setminus Q, M)}$ is refutable and thus $\mathcal{U}_{D_M}(M \setminus Q, M) = (M \setminus Q, M)$. Since $Q \neq \emptyset$, we find that (M, M) is not the least fixpoint of $\mathcal{U}_{D_M}$. Contradiction. □

The hardness reduction in this proof makes use of a statement z that is false in any ultimate two-valued stable model. This can be used to show the same hardness for the credulous reasoning problem for this semantics: we introduce a new statement x that behaves just like $\neg z$, then x is true in some model if and only if there exists a model.

Proposition 4.40. *The problem* $\mathsf{Cred}^{\mathcal{U}}_{stm}$ *is* Σ^P_2*-complete.*

Proof. in Σ^P_2: *Let D be an ADF and $s \in S$. We can guess a set $X \subseteq S$ with $s \in X$ and verify in* coNP *that it is an ultimate two-valued stable model.*

Σ^P_2-hard: *Let $\exists P \forall Q \psi$ be a QBF. We use the same ADF construction as in the hardness proof of* $\mathsf{Exists}^{\mathcal{U}}_{stm}$ *and augment D by an additional statement x with $\varphi_x = \neg z$. It is clear that in any model of D, z must be false and so x must be true. So x is true in some two-valued stable model of D iff D has a two-valued stable model iff $\exists P \forall Q \psi$ is true.* □

A similar argument works for the sceptical reasoning problem: Given a QBF $\forall P \exists Q \psi$, we construct its negation $\exists P \forall Q \neg \psi$, whose associated ADF D has an ultimate two-valued stable model (where z is false) iff $\exists P \forall Q \neg \psi$ is true iff the original QBF $\forall P \exists Q \psi$ is false. Hence $\forall P \exists Q \psi$ is true iff z is true in all ultimate two-valued stable models of D.

Proposition 4.41. *The problem* $\mathsf{Scep}^{\mathcal{U}}_{stm}$ *is* Π^P_2*-complete.*

Proof. in Π^P_2: *Let D be an ADF and $s \in S$. To decide the co-problem, we guess a set $X \subseteq S$ with $s \notin X$ and verify in* coNP *that it is an ultimate two-valued stable model.*

Π^P_2-hard: *Let $\forall P \exists Q \psi$ be a QBF with ψ in CNF. Define the QBF $\exists P \forall Q \neg \psi$ and observe that $\neg \psi$ can be transformed into DNF in linear time. We use this new QBF to construct an ADF D as we did in the hardness proof of* $\mathsf{Exists}^{\mathcal{U}}_{stm}$*. As observed in the proof of Proposition 4.40, the special statement z is false in all ultimate two-valued stable models of D. To show that $\forall P \exists Q \psi$ is true iff z is true in all ultimate two-valued stable models of D, we show that $\forall P \exists Q \psi$ is false iff D has an ultimate two-valued stable model where z is false:*

$\forall P \exists Q \psi$ is false
iff $\neg \forall P \exists Q \psi$ is true
iff $\exists P \forall Q \neg \psi$ is true
iff D has an ultimate two-valued stable model
iff D has an ultimate two-valued stable model where z is false. □

4.2.4 Overview

Table 4.1 below provides a concise overview over the complexity of abstract dialectical frameworks.

approximate ($\mathcal{G}$), σ	conflict-free	naive	admissible	complete	preferred	grounded	model	stable model
$\mathsf{Ver}_\sigma^\mathcal{G}$	NP-c (Proposition 4.11)	DP-c (Proposition 4.17)	coNP-c (Proposition 4.21)	DP-c (Proposition 4.22)	Π_2^p-c (Proposition 4.29)	DP-c (Theorem 4.23)	in P (Brewka et al., 2013, Prop. 5)	in P (Theorem 4.35)
$\mathsf{Exists}_\sigma^\mathcal{G}$	in P (Proposition 4.19)	in P (Corollary 4.20)	Σ_2^p-c (Theorem 4.24)	Σ_2^p-c (Corollary 4.27)	Σ_2^p-c (Corollary 4.27)	coNP-c (Proposition 4.28)	NP-c (Brewka et al., 2013, Prop. 5)	NP-c (Theorem 4.35)
$\mathsf{Cred}_\sigma^\mathcal{G}$	NP-c (Proposition 4.15)	NP-c (Corollary 4.16)	Σ_2^p-c (Proposition 4.30)	Σ_2^p-c (Proposition 4.30, Lemma 4.26)	Σ_2^p-c (Proposition 4.30, Lemma 4.26)	coNP-c (Proposition 4.31)	NP-c (Corollary 4.37)	NP-c (Corollary 4.38)
$\mathsf{Scep}_\sigma^\mathcal{G}$	trivial	Π_2^p-c (Proposition 4.18)	trivial	coNP-c (Corollary 4.32)	Π_3^p-c (Theorem 4.33)	coNP-c (Proposition 4.31)	coNP-c (Corollary 4.37)	coNP-c (Corollary 4.38)
ultimate ($\mathcal{U}$), σ	conflict-free	naive	admissible	complete	preferred	grounded	model	stable model
$\mathsf{Ver}_\sigma^\mathcal{U}$	NP-c (Proposition 4.11)	DP-c (Proposition 4.17)	coNP-c (Brewka et al., 2013, Prop. 10)	DP-c (Brewka et al., 2013, Cor. 7)	Π_2^p-c (Wallner, 2014, Proposition 4.1.14)	DP-c (Brewka et al., 2013, Thm. 6)	in P (Brewka et al., 2013, Prop. 5)	coNP-c (Proposition 4.36)
$\mathsf{Exists}_\sigma^\mathcal{U}$	NP-c (Proposition 4.12)	NP-c (Corollary 4.14)	Σ_2^p-c (Theorem 4.24)	Σ_2^p-c (Corollary 4.27)	Σ_2^p-c (Corollary 4.27)	coNP-c (Proposition 4.28)	NP-c (Brewka et al., 2013, Prop. 5)	Σ_2^p-c (Theorem 4.39)
$\mathsf{Cred}_\sigma^\mathcal{U}$	NP-c (Proposition 4.15)	NP-c (Corollary 4.16)	Σ_2^p-c (Wallner, 2014, Proposition 4.1.11)	Σ_2^p-c (Proposition 4.30, Lemma 4.26)	Σ_2^p-c (Proposition 4.30, Lemma 4.26)	coNP-c (Proposition 4.31)	NP-c (Corollary 4.37)	Σ_2^p-c (Proposition 4.40)
$\mathsf{Scep}_\sigma^\mathcal{U}$	trivial	Π_2^p-c (Proposition 4.18)	trivial	coNP-c (Corollary 4.32)	Π_3^p-c (Wallner, 2014, Theorem 4.1.17)	coNP-c (Wallner, 2014, Proposition 4.1.3)	coNP-c (Corollary 4.37)	Π_2^p-c (Proposition 4.41)

Table 4.1: Complexity results for semantics of Abstract Dialectical Frameworks.

4.3 Bipolar ADFs

In this section, we take a closer look at the special class of ADFs where all links are supporting or attacking, and more importantly the specific link type is known for each link. We first note that since BADFs are a subclass of ADFs, all membership results from the previous sections immediately carry over. However, we can show that many problems will in fact become easier. Intuitively, computing the revision operators is now P-easy because the associated satisfiability/tautology problems only have to treat restricted acceptance formulas. In bipolar ADFs, by definition, if in some three-valued pair (X, Y) a statement s is accepted by a revision operator ($s \in \mathcal{O}'(X, Y)$), it will stay so if we set its undecided supporters to false and its undecided attackers to true. Symmetrically, if a statement is rejected by an operator ($s \notin \mathcal{O}''(X, Y)$), it will stay so if we set its undecided supporters to true and its undecided attackers to false. Hence to decide whether $s \in \mathcal{O}'(X, Y)$ or $s \notin \mathcal{O}''(X, Y)$ for given operator $\mathcal{O}$, pair (X, Y) and statement s, we need only look at *one single interpretation* that can be constructed from the known link types. This is the key idea underlying the next result. Recall that $\mathcal{BG}$ and $\mathcal{BU}$ are the restrictions of the sets of operators $\mathcal{G}$ and $\mathcal{U}$ respectively to BADFs where the type of each link is known.

Proposition 4.42. *Let $\mathcal{I} \in \{\mathcal{BG}, \mathcal{BU}\}$.*

1. $\mathsf{Elem}^{\mathcal{I}'}$ *is in* P.
2. $\mathsf{Elem}^{\mathcal{I}''}$ *is in* P.

Proof. Let D be a BADF with $L = L^+ \dot\cup L^-$ (that is, all links are either *attacking or supporting), $\mathcal{O} \in \{\mathcal{G}_D, \mathcal{U}_D\}$, $s \in S$ and $X \subseteq Y \subseteq S$. The restriction to bipolar ADFs without redundant statements is immaterial as such links can be removed and redundant statements be replaced by an arbitrary truth value constant in the acceptance condition.*

It suffices to show the claims for $\mathcal{I} = \mathcal{BU}$, as Lemma 4.1 tells us that $s \in \mathcal{U}''_D(X, Y)$ iff $s \in \mathcal{G}''_D(X, Y)$; furthermore, due to Proposition 4.6 we know that deciding $s \in \mathcal{G}'_D(X, Y)$ can be done in P.

The main proof idea is to use the information about link polarities to construct canonical interpretations Z such that deciding the element problem for a three-valued pair (X, Y) can be done by deciding the element problem for the two-valued pair (Z, Z).

1. *Define the canonical interpretation through*

$$Z = X \cup (att_D(s) \cap Y)$$

Clearly, since the pair (Z, Z) is two-valued, the problem $s \in \mathcal{U}'_D(Z, Z)$ can be decided in P. *It remains to show that this answers the right question, that is, that $s \in \mathcal{U}'_D(Z, Z)$ iff $s \in \mathcal{U}'_D(X, Y)$.*

if: *Let $s \in \mathcal{U}'_D(X, Y)$. Then $(X, Y) \leq_i (Z, Z)$ implies that $\mathcal{U}_D(X, Y) \leq_i \mathcal{U}_D(Z, Z)$ whence $s \in \mathcal{U}'_D(X, Y) \subseteq \mathcal{U}'_D(Z, Z)$.*

only if: *Let $s \in \mathcal{U}'_D(Z, Z)$. Then $Z \models \varphi_s$ and also $Z \models \varphi_s^{(X,Y)}$.*

Assume to the contrary of what we want to show that $s \notin \mathcal{U}'_D(X, Y)$. Then $\varphi_s^{(X,Y)}$ is refutable and there is a $Z_1 \subseteq S$ with $X \subseteq Z_1 \subseteq Y$ such that $Z_1 \not\models \varphi_s^{(X,Y)}$. Define

$$Z_2 = Z_1 \cup (att_D(s) \cap Y)$$

Clearly $Z_2 \not\models \varphi_s^{(X,Y)}$ since all statements in $att_D(s) \cap Y$ are attacking. Furthermore $X \subseteq Z_1$ implies by definition that $Z \subseteq Z_2$. It also follows that $Z_2 \setminus Z \subseteq sup_D(s)$. Thus since $Z \models \varphi_s^{(X,Y)}$, we conclude that $Z_2 \models \varphi_s^{(X,Y)}$. Contradiction. Therefore $s \in \mathcal{U}'_D(X, Y)$.

2. *Analogously, we define* $Z = X \cup (sup_D(s) \cap Y)$ *and obtain that* $s \in \mathcal{U}''_D(X,Y)$ *iff* $s \in \mathcal{U}''_D(Z,Z)$*: The "if" direction is clear, and for the "only if" direction assume to the contrary that* $s \in \mathcal{U}''_D(X,Y)$ *but* $s \notin \mathcal{U}''_D(Z,Z)$*. By this presumption, the formula* $\varphi^{(X,Y)}$ *is satisfiable, but* $Z \not\models \varphi^{(X,Y)}$*. Let* $Z_1 \subseteq S$ *with* $X \subseteq Z_1 \subseteq Y$ *be such that* $Z_1 \models \varphi^{(X,Y)}$*. Now define*

$$Z_2 = Z_1 \cup (sup_D(s) \cap Y)$$

$X \subseteq Z_1$ implies $Z \subseteq Z_2$ and $Z_2 \setminus Z \subseteq att_D(s)$. Thus $Z \not\models \varphi^{(X,Y)}$ yields $Z_2 \not\models \varphi^{(X,Y)}$. On the other hand, $Z_1 \models \varphi^{(X,Y)}$ and $Z_2 \setminus Z_1 \subseteq sup_D(s)$ whence $Z_2 \models \varphi^{(X,Y)}$. Contradiction. □

Using the generic upper bounds of Theorem 4.8, it is now straightforward to show membership results for BADFs with known link types.

Corollary 4.43. *Let* $\mathcal{I} \in \{\mathcal{BG}, \mathcal{BU}\}$*, semantics* $\sigma \in \{adm, com\}$ *and* $\tau \in \{scf, nai\}$*. We find that*

- $\mathsf{Ver}^{\mathcal{I}}_{\sigma}$, $\mathsf{Ver}^{\mathcal{I}}_{\tau}$ *and* $\mathsf{Ver}^{\mathcal{I}}_{grd}$ *are in* P*;*
- $\mathsf{Ver}^{\mathcal{I}}_{pre}$ *is in* coNP*;*
- $\mathsf{Exists}^{\mathcal{I}}_{\sigma}$, $\mathsf{Exists}^{\mathcal{I}}_{pre}$ *are in* NP*;*
- $\mathsf{Cred}^{\mathcal{I}}_{\tau}$ *is in* P*;*
- $\mathsf{Cred}^{\mathcal{I}}_{\sigma}$ *and* $\mathsf{Cred}^{\mathcal{I}}_{pre}$ *are in* NP*;*
- $\mathsf{Exists}^{\mathcal{I}}_{grd}$, $\mathsf{Cred}^{\mathcal{I}}_{grd}$, $\mathsf{Scep}^{\mathcal{I}}_{grd}$, $\mathsf{Scep}^{\mathcal{I}}_{com}$ *are in* P*;*
- $\mathsf{Scep}^{\mathcal{I}}_{pre}$ *is in* Π^P_2*.*

Proof. Membership is due to Theorem 4.8 and the complexity bounds of the operators in BADFs in Proposition 4.42, just note that $\Sigma^P_0 = \Pi^P_0 = \mathrm{D}^P_0 = \mathsf{P}$*.* $\mathsf{Ver}^{\mathcal{I}}_{grd}$ *is in* $\mathsf{P}^{\mathsf{P}} = \mathsf{P}$ *by Corollary 4.9. For the existence of non-trivial pairs we can simply guess and check in polynomial time for admissible pairs and thus also for complete and preferred semantics.* □

Hardness results straightforwardly carry over from AFs.

Proposition 4.44. *Let* $\mathcal{I} \in \{\mathcal{BG}, \mathcal{BU}\}$ *and semantics* $\sigma \in \{adm, com, pre\}$*.*

- $\mathsf{Ver}^{\mathcal{I}}_{pre}$ *is* coNP*-hard;*
- $\mathsf{Exists}^{\mathcal{I}}_{\sigma}$ *and* $\mathsf{Cred}^{\mathcal{I}}_{\sigma}$ *are* NP*-hard;*
- $\mathsf{Scep}^{\mathcal{I}}_{pre}$ *is* Π^P_2*-hard.*

Proof. Hardness results from AFs for these problems carry over to BADFs as for all semantics AFs are a special case of BADFs. The complexities of the problems on AFs for admissible and preferred semantics were shown by Dimopoulos and Torres (1996), except for the Π^P_2*-completeness result of sceptical preferred semantics, which was shown by Dunne and Bench-Capon (2002). The complete semantics was studied by Coste-Marquis, Devred, and Marquis (2005).* □

4.3.1 Symmetric Conflict-free Semantics

For the semantics based on (symmetric) conflict-freeness, it also becomes P-easy to decide whether non-trivial interpretations exist. Recall that by Lemma 3.48, any set of symmetric conflict-free interpretations is $\leq_i$-downward-closed. (That is, whenever (X, Y) is conflict-free then any $(X', Y') \leq_i (X, Y)$ is also conflict-free.) This also gives a more intuitive explanation of why $\mathsf{Ver}^{\mathcal{I}}_{nai}$ is in P for $\mathcal{I} \in \{\mathcal{BG}, \mathcal{BU}\}$: To verify that a conflict-free pair (X, Y) is also naive, we have to verify that the set of pairs

$$\{(X \cup \{s\}, Y), (X, Y \setminus \{s\}) \mid s \in Y \setminus X\}$$

contains no conflict-free pair. This check can be done in polynomial time since there are at most $2 \cdot |S|$ elements in this set and $\mathsf{Ver}^{\mathcal{I}}_{scf}$ is in P.

Proposition 4.45. *Let $\mathcal{I} \in \{\mathcal{BG}, \mathcal{BU}\}$. $\mathsf{Exists}^{\mathcal{I}}_{scf}$ and $\mathsf{Exists}^{\mathcal{I}}_{nai}$ are in* P.

Proof. We first note that the two decision problems coincide by Lemma 4.13. To decide $\mathsf{Exists}^{\mathcal{I}}_{scf}$ for a given ADF $D = (S, L, C)$, we have to check for each $s \in S$ whether any of the pairs $(\{s\}, S)$ or $(\emptyset, S \setminus \{s\})$ is conflict-free, which can be done in polynomial time by Corollary 4.43. If one of these pairs is conflict-free, the answer is yes; if all pairs where exactly one statement is not undecided are not conflict-free, then there is no non-trivial conflict-free pair. (If there was one, then by Lemma 3.48 there would be a non-trivial conflict-free pair where exactly one statement is true or false.) □

For sceptical reasoning amongst naive semantics, we can show that the problem remains hard even for bipolar ADFs. This is because we can introduce new statements, which allows us to encode tautology checking of propositional formulas in disjunctive normal form into a bipolar ADF.

Proposition 4.46. *Let $\mathcal{I} \in \{\mathcal{BG}, \mathcal{BU}\}$. $\mathsf{Scep}^{\mathcal{I}}_{nai}$ is* coNP*-complete.*

Proof. in coNP: *To verify that a statement $s \in S$ does not follow sceptically, we can guess a pair (X, Y) with $s \notin X$ and verify in* P *that it is naive.*

coNP-hard: *We reduce from tautology checking. Let $\psi = \psi_1 \vee \ldots \vee \psi_n$ be a propositional formula in DNF over vocabulary P. Assume additionally (and without loss of generality) that there is no disjunct ψ_i that contains both p and $\neg p$ for some $p \in P$. (If there is such a disjunct, we can remove it without changing the models of ψ.) We construct a bipolar ADF $D = (S, L, C)$ as follows:*

$$\begin{aligned} S &= P \cup \{z, d_1, \ldots, d_n\} \\ \varphi_p &= p && (p \in P) \\ \varphi_{d_i} &= \psi_i && (1 \leq i \leq n) \\ \varphi_z &= d_1 \vee \ldots \vee d_n \end{aligned}$$

We show that z is contained in all naive pairs of D iff ψ is a tautology.

"if": *Let ψ be a tautology. Given an $M \subseteq P$, define a set N_M as follows:*

$$N_M = M \cup \{d_i \mid M \models \psi_i\} \cup \{z\}$$

We show that for each $M \subseteq P$, the pair $\bar{m}_M = (N_M, N_M)$ is naive, and these are the only naive pairs. We first observe that each such pair is two-valued, and thus the two operators

(approximate and ultimate) coincide on it, furthermore we need only show conflict-freeness to show naivety. It is clear that $\bar{m}_M$ is conflict-free with respect to all $p \in P$. For $1 \leq i \leq n$, conflict-freeness of $\bar{m}_M$ with respect to d_i follows by definition. Since ψ is a tautology, there is at least one d_i in each N_M, and $z \in N_M$ is justified. Assume there were another naive pair (X, Y) with $z \notin X$. First of all, each naive pair must constitute a two-valued interpretation of the statements in P, for otherwise the $\leq_i$-maximality condition would be violated. Now this enforces a fixed truth value for $d_1, \ldots, d_n$ and thus also for z. As argued above, $z \in N_M$ necessarily holds.

"only if": *Let ψ be refutable. Then there is an $M \subseteq P$ such that we find $M \not\models \psi_i$ for all $1 \leq i \leq n$. We show that the pair $\bar{m} = (M, M)$ is naive for approximate and ultimate operator. Clearly by presumption, for all $1 \leq i \leq n$ we find that $\psi_i^{(M,M)}$ is a Boolean expression that evaluates to false, so having $d_i \notin M$ in the upper bound of the pair $\bar{m}$ is justified. Finally, $\varphi_z = d_1 \vee \ldots \vee d_n$ also evaluates to false, thus justifying $z \notin M$. Thus there is a naive pair $(X, Y) = (M, M)$ with $z \notin X$.* □

Notably, this result is the only case in which bipolar ADFs are (potentially) more complex than AFs, as in the latter sceptical reasoning over naive pairs can be done in polynomial time (Coste-Marquis et al., 2005).[3]

4.3.2 Two-valued Semantics

Regarding BADFs and two-valued semantics we first show that there is no hope that the existence problems for approximate and ultimate two-valued stable models coincide as there are cases when the semantics differ.

Example 4.4. Consider the BADF $D = (S, L, C)$ with statements $S = \{a, b, c\}$ and acceptance formulas $\varphi_a = \top$, $\varphi_b = a \vee c$ and $\varphi_c = a \vee b$. The only two-valued supported model is (S, S) where all statements are true. This pair is also an ultimate two-valued stable model, since $\mathcal{U}'_D(\emptyset, S) = \{a\}$, and both $\varphi_b^{(\{a\},S)} = \top \vee c$ and $\varphi_c^{(\{a\},S)} = \top \vee b$ are tautologies, whence we have $\mathcal{U}'_D(\{a\}, S) = S$. However, (S, S) is not an approximate two-valued stable model: although $\mathcal{G}'_D(\emptyset, S) = \{a\}$, then $\mathcal{G}'_D(\{a\}, S) = \{a\}$ since the partially evaluated formulas $\varphi_b^{(\{a\},S)}$ and $\varphi_c^{(\{a\},S)}$ contain propositional variables. We thus cannot reconstruct the upper bound S and D has no approximate two-valued stable models. ◇

So approximate and ultimate two-valued stable model semantics are indeed different. However, we can show that the respective existence problems have the same complexity.

Proposition 4.47. *Let $\mathcal{I} \in \{\mathcal{BG}, \mathcal{BU}\}$ and semantics $\sigma \in \{mod, stm\}$. $\mathsf{Ver}_\sigma^\mathcal{I}$ is in P; $\mathsf{Exists}_\sigma^\mathcal{I}$ is NP-complete.*

Proof. Membership carries over – for supported models from Proposition 5 of Brewka et al. (2013), for approximate stable models from Theorem 4.35. For membership for ultimate stable models, we can use Proposition 4.42 to adapt the decision procedure of Proposition 4.34. In any case, hardness carries over from AFs (Dimopoulos and Torres, 1996). □

[3] To check whether an argument a is sceptically accepted for naive semantics, we only have to check whether all its attackers are self-attacking: if there is a b that attacks a and is not self-attacking, then the set $\{b\}$ is conflict-free, thus there exists a naive set $N \supseteq \{b\}$ with $a \notin N$.

For credulous and sceptical reasoning over the two-valued semantics, membership is straightforward and hardness again carries over from argumentation frameworks.

Corollary 4.48. *Let $\mathcal{I} \in \{\mathcal{BG}, \mathcal{BU}\}$ and semantics $\sigma \in \{mod, stm\}$.* $\mathsf{Cred}_\sigma^{\mathcal{I}}$ *is* NP*-complete;* $\mathsf{Scep}_\sigma^{\mathcal{I}}$ *is* coNP*-complete.*

4.3.3 Overview

$\mathcal{I} \in \{\mathcal{BG}, \mathcal{BU}\}, \sigma$	conflict-free	naive	admissible	complete	preferred	grounded	model	stable model
$\mathsf{Ver}_\sigma^\mathcal{I}$	in P (Corollary 4.43)	in P (Corollary 4.43)	in P (Corollary 4.43)	in P (Corollary 4.43)	coNP-c (Corollary 4.43, Proposition 4.44)	in P (Corollary 4.43)	in P (Proposition 4.47)	in P (Proposition 4.47)
$\mathsf{Exists}_\sigma^\mathcal{I}$	in P (Proposition 4.45)	in P (Proposition 4.45)	NP-c (Corollary 4.43, Proposition 4.44)	NP-c (Corollary 4.43, Proposition 4.44)	NP-c (Corollary 4.43, Proposition 4.44)	in P (Corollary 4.43)	NP-c (Proposition 4.47)	NP-c (Proposition 4.47)
$\mathsf{Cred}_\sigma^\mathcal{I}$	in P (Corollary 4.43)	in P (Corollary 4.43)	NP-c (Corollary 4.43, Proposition 4.44)	NP-c (Corollary 4.43, Proposition 4.44)	NP-c (Corollary 4.43, Proposition 4.44)	in P (Corollary 4.43)	NP-c (Corollary 4.48)	NP-c (Corollary 4.48)
$\mathsf{Scep}_\sigma^\mathcal{I}$	trivial	coNP-c (Proposition 4.46)	trivial	in P (Corollary 4.43)	Π_2^P-c (Corollary 4.43, Proposition 4.44)	in P (Corollary 4.43)	coNP-c (Corollary 4.48)	coNP-c (Corollary 4.48)

Table 4.2: Complexity results for semantics of bipolar Abstract Dialectical Frameworks.

4.4 Conclusion

In this chapter we studied the computational complexity of abstract dialectical frameworks using approximation fixpoint theory. We showed numerous novel results for two families of ADF semantics, the approximate and ultimate semantics, which are themselves inspired by argumentation and AFT. We showed that in most cases the complexity increases by one level of the polynomial hierarchy compared to the corresponding reasoning tasks on AFs. Notable differences between the two families of semantics lie in the stable model semantics and in semantics based on symmetric conflict-freeness, where the approximate version is easier than its ultimate counterpart. For the restricted, yet powerful class of bipolar ADFs we proved that for the corresponding reasoning tasks AFs and BADFs have (almost) the same complexity, with the single exception of sceptical reasoning among (symmetric) naive pairs. This suggests that many types of relations between arguments can be introduced without increasing the worst-time complexity. On the other hand, our results again emphasise that arbitrary (non-bipolar) ADFs cannot be compiled into equivalent Dung AFs in deterministic polynomial time, unless the polynomial hierarchy collapses to the first level. Under the same assumption, ADFs cannot be implemented directly with methods that are typically applied to AFs, for example answer-set programming (Egly et al., 2010).

Our results on the complexity of bipolar ADFs led to our extending the ADF system DIAMOND (Ellmauthaler and Strass, 2014, 2016) with specialised implementation techniques for bipolar ADFs. In the future, we also plan to accommodate the approximate semantics family into DIAMOND. In another direction of work, QBF encodings for general ADFs were developed and implemented in the system QADF (Diller, Wallner, and Woltran, 2015). For further future work several promising directions are possible. Studying easier fragments of ADFs as well as parameterised complexity analysis can lead to efficient decision procedures, as is witnessed for AFs (Dvořák, Järvisalo, Wallner, and Woltran, 2014; Dvořák, Ordyniak, and Szeider, 2012). We also deem it auspicious to use alternative representations of acceptance conditions, for instance by employing techniques from knowledge compilation (Darwiche and Marquis, 2002).

In recent related work, Gaggl et al. (2015) analysed the computational complexity of naive-based ADF semantics as defined by Gaggl and Strass (2014), that is, based on the asymmetric version of conflict-freeness applied to the ultimate approximation operator. A detailed comparison of the two types of semantics and their respective complexities is left for future work. A complexity analysis of other useful AF semantics would also reveal further insights, for example semi-stable semantics (Verheij, 1996; Caminada, Carnielli, and Dunne, 2012) or ideal semantics (Dung, Mancarella, and Toni, 2007; Dunne, 2009; Booth, 2015). Furthermore, Polberg, Wallner, and Woltran (2013) and Polberg (2014, 2016) proposed several extension-based semantics for ADFs, and a complexity analysis would be interesting.

For semantical analysis, it would be useful to consider principle-based evaluations for ADFs (Baroni and Giacomin, 2007). Furthermore it appears natural to compare (ultimate) ADF semantics and ultimate logic programming semantics (Denecker et al., 2004) in approximation fixpoint theory, in particular with respect to computational complexity. Finally, we could apply the general operator splitting results of Vennekens, Gilis, and Denecker (2006) to abstract argumentation and compare them to the stand-alone results obtained for AFs (Baumann, 2011) and ADFs (Linsbichler, 2014).

Chapter 5

Relative Expressiveness and Succinctness

More often than not, different knowledge representation languages have conceptually similar and partially overlapping intended application areas. What are we to do if faced with an application and a choice of several possible knowledge representation languages that could be used for the application? One of the first axes along which to compare different formalisms that comes to mind is computational complexity: if a language is computationally too expensive when considering the problem sizes typically encountered in practice, then this is a clear criterion for exclusion. But what if the available language candidates have the same computational complexity? If their expressiveness in the computational-complexity sense of "What kinds of *problems* can the formalism solve?" is the same, we need a more fine-grained notion of expressiveness. In this chapter, we use such a notion and study the expressiveness of abstract dialectical frameworks, the generalisation of abstract argumentation frameworks that is the main object of study of this thesis.

Argumentation frameworks are the de-facto standard formalism in abstract argumentation, a field that studies how (abstract) arguments relate to each other in terms of directed conflicts ("attacks"), and how these conflicts can be resolved without "looking into" the arguments. While AFs are popular and well-studied, it has been noted many times in the literature that their expressive capabilities are somewhat limited. This has only recently been made technically precise by Dunne, Dvořák, Linsbichler, and Woltran (2014, 2015), who basically showed that introducing new, purely technical arguments is sometimes inevitable when using AFs for knowledge representation purposes. However, due to their very nature, the dialectical meaning of such technical arguments might be – ironically – debatable.

Not surprisingly, quite a number of generalisations of AFs have been proposed (for an overview we refer to the work of Brewka et al., 2014). As one of the most general AF alternatives, ADFs have emerged. ADFs could be called the lovechild of AFs and logic programs, since they combine intuitions and semantics from Dung-style abstract argumentation as well as logic programming (Brewka et al., 2013; Alviano and Faber, 2015; see also Chapter 3 of this thesis). While on the abstract level, ADFs are intended to function as "argumentation middleware" – a sufficiently expressive target formalism for translations from more concrete (application) formalisms.

In this chapter, we approach abstract dialectical frameworks as knowledge representation formalisms, since they are used to represent knowledge about arguments and relationships between these arguments. We employ this view to analyse the representational capabilities

of ADFs. Due to their roots in AFs and normal logic programs, we also compare the representational capabilities of those formalisms in the same setting. In this initial study we restrict ourselves to looking at two-valued semantics, more specifically the ADF (stable) model semantics, which corresponds to AF stable extension semantics, and the supported and stable model semantics for normal logic programs. We add propositional logic to have a well-known reference point. Analysing these precise formalisms additionally makes sense to us because the computational complexity of their respective model existence problems is the same (with one exception):

- for AFs, deciding stable extension existence is NP-complete (Dimopoulos et al., 2002);
- for normal logic programs, deciding the existence of supported/stable models is NP-complete (Bidoit and Froidevaux, 1991; Marek and Truszczyński, 1991);
- for ADFs, deciding the existence of (supported) models is NP-complete, deciding the existence of stable models is Σ_2^P-complete for general ADFs and NP-complete for the subclass of bipolar ADFs (Chapter 4);
- the propositional satisfiability problem is NP-complete (Papadimitriou, 2003; Arora and Barak, 2009).

In view of these almost identical complexities, we use an alternative measure of the expressiveness of a knowledge representation formalism $\mathcal{F}$: "Given a set of two-valued interpretations, is there a knowledge base in $\mathcal{F}$ that has this exact model set?" This notion has been introduced by Gogic, Kautz, Papadimitriou, and Selman (1995) and lends itself straightforwardly to compare different formalisms:

> Formalism $\mathcal{F}_2$ is at least as expressive as formalism $\mathcal{F}_1$ if and only if every knowledge base in $\mathcal{F}_1$ has an equivalent knowledge base in $\mathcal{F}_2$.

So here expressiveness is understood in terms of *realisability*, "What kinds of model sets can the formalism express?" (In model theory, this is known as *definability*.)

It is easy to see that propositional logic can express any set of two-valued interpretations, it is *universally expressive*. The same is easy (but less easy) to see for normal logic programs under supported model semantics. For normal logic programs (NLPs) under *stable* model semantics, it is clear that not all model sets can be expressed, since two different stable models are always incomparable with respect to the subset relation. For canonical logic programs (CLPs, where we allow nested expressions of the form "*not not p*" in rule bodies; Lifschitz, Tang, and Turner, 1999), expressiveness again jumps up to all interpretation sets (Lifschitz and Razborov, 2006). In this chapter, we study such expressiveness properties for all the mentioned formalisms under different semantics. It turns out that the languages form a more or less strict expressiveness hierarchy, with AFs at the bottom, ADFs and normal LPs under stable semantics higher up and ADFs, normal LPs under supported model semantics, and canonical LPs at the top together with propositional logic.

To show that a language $\mathcal{F}_2$ is at least as expressive as a language $\mathcal{F}_1$ we will mainly use two different techniques. In the best case, we can use a syntactic compact and faithful translation from knowledge bases of $\mathcal{F}_1$ to those of $\mathcal{F}_2$. *Compact* means that the translation does not change the vocabulary, that is, does not introduce new atoms. *Faithful* means that the translation exactly preserves the models of the knowledge base for respective semantics of the two languages. In the second best case, we assume the knowledge base of $\mathcal{F}_1$ to be given in the form of a set X of desired models and construct a semantic *realisation* of X in $\mathcal{F}_2$, that is, a knowledge base in $\mathcal{F}_2$ with model set precisely X. To show that language $\mathcal{F}_2$ is *strictly more*

expressive than $\mathcal{F}_1$, we additionally have to present a knowledge base kb from $\mathcal{F}_2$ of which we prove that $\mathcal{F}_1$ cannot express the model set of kb.

Analysing the expressiveness of argumentation formalisms is a quite recent strand of work. Its ascent can be attributed to Dunne et al. (2014, 2015), who studied realisability for argumentation frameworks (allowing to introduce new arguments as long as they are never accepted). Likewise, Dyrkolbotn (2014) analysed AF realisability under projection (allowing to introduce new arguments) for three-valued semantics. Baumann et al. (2014, 2016) studied the expressiveness of the subclass of "compact" AFs, where each argument is accepted at least once. Finally, Pührer (2015) analysed the realisability of three-valued semantics for ADFs. Previous more preliminary works include that of Brewka et al. (2011), who translated ADFs into AFs for the ADF model and AF stable extension semantics, albeit that translation introduces additional arguments and is therefore not compact.

The gain that is achieved by our analysis in this chapter is not only one of increased clarity about fundamental properties of these knowledge representation languages – *What can these formalisms express, actually?* – but has several further applications. As Dunne et al. (2015) remarked, a major application is in constructing knowledge bases with the aim of encoding a certain model set. As a necessary prerequisite to this, it must be known that the intended model set is realisable in the first place. For example, in a recent approach to revising argumentation frameworks (Coste-Marquis, Konieczny, Mailly, and Marquis, 2014), the authors avoid this problem by assuming to produce a *collection* of AFs whose model sets in union produce the desired model set. While the work of Dunne et al. (2015) showed that this is indeed necessary in the case of AFs and stable extension semantics, our work shows that for ADFs under the model semantics, a single knowledge base (ADF) is always enough to realise any given model set. What is more, if we assume that the intended model set is given in the form of a propositional formula, then the size of the realising ADF is at most linear in the size of the formula. This is only one example – we will on several occasions also consider the sizes of realisations, as is not uncommon in logic-based AI (Darwiche and Marquis, 2002; Lifschitz and Razborov, 2006; French, van der Hoek, Iliev, and Kooi, 2013; Shen and Zhao, 2014). Indeed, representation size is a fundamental practical aspect of knowledge representation languages: universal expressiveness is of little use if the model sets to express require exponential-size knowledge bases even in the best case!

Of course, the fact that the languages we study have the same computational complexity means that there in principle exist polynomial intertranslations for the respective decision problems. But such intertranslations may involve the introduction of a polynomial number of new atoms. In theory, an increase from n atoms to n^k atoms for some $k > 1$ is of no consequence. In practice, it has a profound impact: the number n of atoms directly influences the search space that any implementation potentially has to cover. There, a step from 2^n to

$$2^{n^k} = 2^{n^{k-1}n} = \left(2^{n^{k-1}}\right)^n$$

amounts to an *exponential* increase in search space size. Being able to realise a model set compactly, without new atoms, therefore attests that a formalism $\mathcal{F}$ has a certain basic kind of efficiency property, in the sense that the $\mathcal{F}$-realisation of a model set does not unnecessarily enlarge the search space of algorithms operating on it.

It might seem that it is a restricting assumption to view formalisms as sets $\mathcal{F}$ of knowledge bases kb where $\mathcal{F}$ is associated with a two-valued semantics. However, this language representation model is universal in the sense that it is just another way of expressing languages as sets of words over $\{0,1\}$. Using an n-element vocabulary $A_n = \{a_1, \ldots, a_n\}$, a binary word $w = x_1 x_2 \cdots x_n$ of length n is encoded as the set $M_w = \{a_i \in A_n \mid x_i = 1\} \subseteq A_n$.

For example, using the vocabulary $A_3 = \{a_1, a_2, a_3\}$, the binary word 101 of length 3 corresponds to the set $M_{101} = \{a_1, a_3\}$. Consequently, a set L_n of words of length n can be represented by a set $X_{L_n} \subseteq 2^{A_n}$ of subsets of A_n: $X_{L_n} = \{M_w \mid w \in L_n\}$. With the above example vocabulary, the word set $L_3 = \{101, 110, 011\}$ is represented by the model set $X_{L_3} = \{\{a_1, a_3\}, \{a_1, a_2\}, \{a_2, a_3\}\}$. Conversely, each sequence $(X_n)_{n\geq 0}$ of sets with $X_n \subseteq 2^{A_n}$ uniquely determines a language $L = \bigcup_{n\geq 0} L_n$ over $\{0, 1\}$: for each $n \in \mathbb{N}$, we have $L_n = \{w_M \mid M \in X_n\}$ with $w_M = x_1 x_2 \cdots x_n$ where for each $i \in \{1, \ldots, n\}$, $x_i = 1$ if $a_i \in M$ and $x_i = 0$ if $a_i \notin M$. In this chapter we use "language" to refer to object-level languages, while "formalism" refers to meta-level languages such as propositional logic, argumentation frameworks, abstract dialectical frameworks, and logic programs.

Formally, the syntax of ADFs is defined via Boolean functions. However, we are interested in representations of ADFs. So we have to fix a representation of ADFs via fixing a representation of Boolean functions. We choose to use (unrestricted) propositional formulas, as is customary in most of the literature (Brewka and Woltran, 2010; Brewka et al., 2013; Polberg et al., 2013; Polberg, 2014; Gaggl and Strass, 2014; Linsbichler, 2014; Strass and Wallner, 2015; Pührer, 2015; Gaggl et al., 2015; Strass, 2015a; Strass, 2015b). Exceptions to this custom are the works of Brewka et al. (2011), who use Boolean circuits, and one of ours (Strass, 2013) where we used characteristic models (that is, used a representation that is equivalent to representing the formulas in disjunctive normal form). For the subclass of bipolar ADFs, yet no uniform representation exists, which is another question we address in this chapter.

By propositional formulas over a vocabulary A we mean formulas over the Boolean basis $\{\wedge, \vee, \neg\}$, that is, trees whose leaves (sinks) are atoms from A or the logical constants true $\top$ or false $\bot$, and internal nodes are either unary ($\neg$) or binary ($\wedge$,$\vee$). We also make occasional use of Boolean circuits, where "trees" above is replaced by "directed acyclic graphs"; in particular, we allow unbounded fan-in, that is, reusing sub-circuits. As usual, the depth of a formula (circuit) is the length of the longest path from the root to a leaf (sink). Figure 5.1 below shows formula and circuit examples of depth 3.

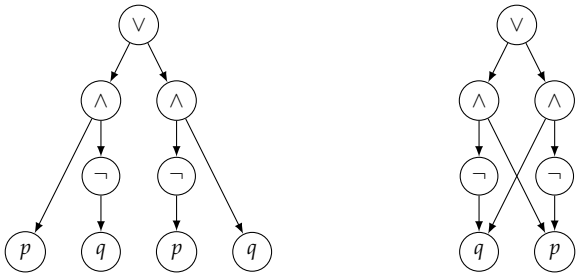


Figure 5.1: Representing $(p \wedge \neg q) \vee (q \wedge \neg p)$ as a formula tree (left) and a circuit (right).

Analysing the expressive power and representation size of Boolean circuits is an established sub-field of computational complexity (Arora and Barak, 2009; Jukna, 2012). This has led to a number of language classes whose members can be recognised by Boolean circuits satisfying certain restrictions. We will need the class AC^0, which contains all languages $L = \bigcup_{n\geq 0} L_n$ for which there exist $d, k \in \mathbb{N}$ such that for each $n \in \mathbb{N}$, there exists a Boolean circuit C_n of depth at most d and size at most n^k where the models of C_n exactly express L_n.[1] In other words, every language $L \in \mathsf{AC}^0$ can be recognised by a family of polynomial-size Boolean circuits of a fixed maximal depth that is independent of word length.

[1] To be more precise, for each $n \in \mathbb{N}$, the models of C_n are exactly X_{L_n}, which in turn expresses L_n.

The chapter proceeds as follows. We first define the notion of expressiveness (and succinctness) formally and then introduce the formalisms we will study. After reviewing several intertranslatability results for these languages, we step-wise obtain the results that lead to the expressiveness hierarchy, while at times also looking at representational efficiency. We finally show that allowing to linearly expand the vocabulary leads to a collapse of the hierarchy. The chapter concludes with a discussion of possible future work.

5.1 Background on Relative Expressiveness

We presume a finite set A of atoms (statements, arguments), the *vocabulary*. A knowledge representation formalism interpreted over A is then some set $\mathcal{F}$; a (two-valued) semantics for $\mathcal{F}$ is a mapping $\sigma : \mathcal{F} \to 2^{2^A}$ that assigns sets of two-valued models to knowledge bases $\mathsf{kb} \in \mathcal{F}$. (So A is implicit in σ.) Strictly speaking, a two-valued interpretation is a mapping from the set of atoms into the two truth values true and false, but for technical ease we represent two-valued interpretations by the sets containing the atoms that are true. Below, we write $\sigma(\mathcal{F}) = \{\sigma(\mathsf{kb}) \mid \mathsf{kb} \in \mathcal{F}\}$; intuitively, $\sigma(\mathcal{F})$ is the set of interpretation sets that formalism $\mathcal{F}$ can express, with any knowledge base whatsoever. For example, for $\mathcal{F} = \mathrm{PL}$ propositional logic and $\sigma = \mathit{mod}$ the usual model semantics, we have $\sigma(\mathrm{PL}) = 2^{2^A}$ since obviously any set of models is realisable in propositional logic.[2] This leads us to compare different pairs of languages and semantics with respect to the semantics' range of models. Our concept of "formalism" concentrates on semantics and decidedly remains abstract. We first define the expressiveness relation among formalisms.

Definition 5.1. Let A be a finite vocabulary, $\mathcal{F}_1, \mathcal{F}_2$ be formalisms that are interpreted over A and $\sigma_1 : \mathcal{F}_1 \to 2^{2^A}$ and $\sigma_2 : \mathcal{F}_2 \to 2^{2^A}$ be two-valued semantics. We define

$$\mathcal{F}_1^{\sigma_1} \leq_e \mathcal{F}_2^{\sigma_2} \quad \text{iff} \quad \sigma_1(\mathcal{F}_1) \subseteq \sigma_2(\mathcal{F}_2) \qquad \diamond$$

Intuitively, formalism $\mathcal{F}_2$ under semantics σ_2 is at least as expressive as formalism $\mathcal{F}_1$ under semantics σ_1, because all model sets that $\mathcal{F}_1$ can express under σ_1 are also contained in those that $\mathcal{F}_2$ can produce under σ_2. (If the semantics are clear from the context we will omit them; this holds in particular for argumentation frameworks and propositional logic, where we only look at a single semantics.) As usual,

- $\mathcal{F}_1 <_e \mathcal{F}_2$ iff $\mathcal{F}_1 \leq_e \mathcal{F}_2$ and $\mathcal{F}_2 \not\leq_e \mathcal{F}_1$;
- $\mathcal{F}_1 \cong_e \mathcal{F}_2$ iff $\mathcal{F}_1 \leq_e \mathcal{F}_2$ and $\mathcal{F}_2 \leq_e \mathcal{F}_1$.

The relation $\leq_e$ is reflexive and transitive by definition, but not necessarily antisymmetric. That is, there might different formalisms $\mathcal{F}_1 \neq \mathcal{F}_2$ that are equally expressive, i.e. $\mathcal{F}_1 \cong_e \mathcal{F}_2$.

We next introduce the succinctness relation as defined by Gogic et al. (1995).

Definition 5.2. Let A be a finite vocabulary; let $\mathcal{F}_1$ and $\mathcal{F}_2$ be formalisms that are interpreted over A, have size measures $\|\cdot\|_1$ and $\|\cdot\|_2$, and two-valued semantics σ_1 and σ_2, respectively. Define $\mathcal{F}_1^{\sigma_1} \leq_s \mathcal{F}_2^{\sigma_2}$ if and only if there is a $k \in \mathbb{N}$ such that for all $\mathsf{kb}_1 \in \mathcal{F}_1$ with $\sigma_1(\mathsf{kb}_1) \in \sigma_1(\mathcal{F}_1) \cap \sigma_2(\mathcal{F}_2)$, there is a $\mathsf{kb}_2 \in \mathcal{F}_2$ with $\sigma_1(\mathsf{kb}_1) = \sigma_2(\mathsf{kb}_2)$ and $\|\mathsf{kb}_2\|_2 \leq \|\mathsf{kb}_1\|_1^k$. $\diamond$

[2] For a set $X \subseteq 2^A$ we can simply define $\varphi_X = \bigvee_{M \in X} \varphi_M$ with $\varphi_M = \bigwedge_{a \in M} a \wedge \bigwedge_{a \in A \setminus M} \neg a$ and clearly $\mathit{mod}(\varphi_X) = X$.

Intuitively, $\mathcal{F}_1^{\sigma_1} \leq_s \mathcal{F}_2^{\sigma_2}$ means that $\mathcal{F}_2$ under σ_2 is at least as succinct as $\mathcal{F}_1$ under σ_1. Put another way, for $\mathcal{F}_1^{\sigma_1} \leq_s \mathcal{F}_2^{\sigma_2}$ to hold, any knowledge base from $\mathcal{F}_1$ with an equivalent counterpart in $\mathcal{F}_2$ must have an equivalent counterpart *that is at most polynomially larger*. Note that succinctness talks only about those model sets that both can express, so it is most meaningful when comparing languages that are equally expressive, that is, whenever $\sigma_1(\mathcal{F}_1) = \sigma_2(\mathcal{F}_2)$. As usual, we define $\mathcal{F}_1 <_s \mathcal{F}_2$ iff $\mathcal{F}_1 \leq_s \mathcal{F}_2$ and $\mathcal{F}_2 \not\leq_s \mathcal{F}_1$, and $\mathcal{F}_1 \cong_s \mathcal{F}_2$ iff $\mathcal{F}_1 \leq_s \mathcal{F}_2$ and $\mathcal{F}_2 \leq_s \mathcal{F}_1$. The relation $\leq_s$ is reflexive, but not necessarily antisymmetric or transitive.

The final general definition is about formalisms polynomially expressing languages. Here, we already make use of the previously introduced bijection between interpretations and binary words and use the term "languages" to synonymously refer to both.

Definition 5.3. A formalism $\mathcal{F}$ can polynomially express a language $L = \bigcup_{n\geq 0} L_n$ under semantics $\sigma : \mathcal{F} \to 2^{2^A}$ if and only if there is a $k \in \mathbb{N}$ such that for each positive $n \in \mathbb{N}$ there is a knowledge base $\mathsf{kb}_n \in \mathcal{F}$ of that formalism such that $\sigma(\mathsf{kb}_n) = L_n$ and $\|\mathsf{kb}_n\| \in O(n^k)$. ◇

We next introduce some specific object-level languages that we will use. First of all, the language Parity contains all odd-element subsets of the vocabulary. Formally, for $A_n = \{a_1, \ldots, a_n\}$ with $n \geq 1$ we have

$$\textsc{Parity}_n = \{M \subseteq A_n \mid \exists m \in \mathbb{N} : |M| = 2m + 1\}$$

As explained before, then $\textsc{Parity} = \bigcup_{n\in\mathbb{N}, n\geq 1} \textsc{Parity}_n$. It is a textbook result that Parity is expressible by polynomial-size propositional formulas (Jukna, 2012); for example, we can define $\Phi_1^{\textsc{Parity}}(a_1) = a_1$ and for $n \geq 2$ set

$$\begin{aligned}\Phi_n^{\textsc{Parity}}(a_1, \ldots, a_n) = {} & (\Phi_{n_\downarrow}^{\textsc{Parity}}(a_1, \ldots, a_{n_\downarrow}) \wedge \neg\Phi_{n_\uparrow}^{\textsc{Parity}}(a_{n_\downarrow+1}, \ldots, a_n)) \vee \\ & (\neg\Phi_{n_\downarrow}^{\textsc{Parity}}(a_1, \ldots, a_{n_\downarrow}) \wedge \Phi_{n_\uparrow}^{\textsc{Parity}}(a_{n_\downarrow+1}, \ldots, a_n))\end{aligned}$$

with $n_\downarrow = \lfloor \frac{n}{2} \rfloor$ and $n_\uparrow = \lceil \frac{n}{2} \rceil$. (This construction yields a formula of logarithmic depth and therefore polynomial size.) It is also a textbook result (although not nearly as easy to see) that Parity cannot be expressed by depth-bounded polynomial-size circuits, that is, $\textsc{Parity} \notin \mathsf{AC}^0$ (Jukna, 2012).

As another important class, threshold languages are defined for $n, k \in \mathbb{N}$ with $n \geq 1$ and $k \leq n$:

$$\textsc{Threshold}_{n,k} = \{M \subseteq A_n \mid k \leq |M|\}$$

That is, $\textsc{Threshold}_{n,k}$ contains all interpretations over n atoms where at least k atoms are true. The special case $k = \lceil \frac{n}{2} \rceil$ leads to the majority languages,

$$\textsc{Majority}_n = \textsc{Threshold}_{n,\lceil \frac{n}{2} \rceil}$$

that contain all interpretations where at least half of the atoms in the vocabulary are true.

5.1.1 Translations Between Considered Formalisms

We briefly review all known translations between the mentioned formalisms.

From AFs to BADFs Brewka and Woltran (2010) showed how to translate AFs into ADFs: For an AF $F = (A, R)$, define the ADF associated to F as $D_F = (A, R, C)$ with $C = \{\varphi_a\}_{a \in A}$ and $\varphi_a = \bigwedge_{(b,a) \in R} \neg b$ for $a \in A$. Clearly, the resulting ADF is bipolar: parents are always attacking. Brewka and Woltran proved that this translation is faithful for the AF stable extension and ADF model semantics (Proposition 1). Brewka et al. (2013) later proved the same for the AF stable extension and ADF stable model semantics (Theorem 4). It is easy to see that the translation can be computed in polynomial time and induces at most a linear blowup.

From ADFs to PL Brewka and Woltran (2010) also showed that ADFs under supported model semantics can be faithfully translated into propositional logic: when acceptance conditions of statements $a \in A$ are represented by propositional formulas φ_a, then the supported models of an ADF D over A are given by the classical propositional models of the formula set $\Phi_D = \{a \leftrightarrow \varphi_a \mid a \in A\}$.

From AFs to PL In combination, the previous two translations yield a polynomial and faithful translation chain from AFs into propositional logic: $\Phi_{(A,R)} = \left\{ a \leftrightarrow \left(\bigwedge_{(b,a) \in R} \neg b \right) \;\middle|\; a \in A \right\}$.

From ADFs to NLPs Earlier in this thesis (Chapter 3), we showed that ADFs can be faithfully translated into normal logic programs. For an ADF $D = (A, L, C)$, its standard NLP is

$$P_D = \{a \leftarrow (M \cup \mathit{not}\,(\mathit{par}(a) \setminus M)) \mid a \in A, C_a(M) = \mathbf{t}\}$$

It follows from Lemma 3.14 (Lemma 3.14 of Strass (2013)) that this translation preserves the supported model semantics. The translation is size-preserving for the acceptance condition representation of Chapter 3 via characteristic models; when representing acceptance conditions via propositional formulas, this cannot be guaranteed as we will show later.[3]

From AFs to NLPs The translation chain from AFs to ADFs to NLPs is compact, and faithful for AF stable semantics and NLP stable semantics (Osorio et al., 2005), and AF stable semantics and NLP supported semantics. It is size-preserving since the single rule for each atom contains all attackers once: $P_{(A,R)} = \{a \leftarrow \{\mathit{not}\ b \mid (b,a) \in R\} \mid a \in A\}$.

From NLPs to PL It is well-known that normal logic programs under supported model semantics can be translated to propositional logic (Clark, 1978). A normal logic program P becomes the propositional theory Φ_P,

$$\Phi_P = \{a \leftrightarrow \varphi_a \mid a \in A\} \quad \text{where} \quad \varphi_a = \bigvee_{a \leftarrow B \in P} \left(\bigwedge_{b \in B^+} b \wedge \bigwedge_{b \in B^-} \neg b \right) \quad \text{for } a \in A.$$

For the stable model semantics, additional formulas have to be added, but the extended translation works all the same (Lin and Zhao, 2004). That translation can even be extended to canonical logic programs (Lee, 2005).

[3] Already for complexity reasons, we cannot expect that this translation is also faithful for the stable semantics. And indeed, the ADF $D = (\{a\}, \{(a,a)\}, \{\varphi_a = a \vee \neg a\})$ has a stable model $\{a\}$ while its standard logic program $P(D) = \{a \leftarrow \{a\}, a \leftarrow \{\mathit{not}\ a\}\}$ has no stable model. However, it holds that $st(P(D)) \subseteq st(D)$ (Denecker et al., 2004; Strass, 2013).

From NLPs to ADFs The Clark completion of a normal logic program directly yields an equivalent ADF over the same signature (Brewka and Woltran, 2010). Clearly the translation is computable in polynomial time and the blowup (with respect to the original logic program) is at most linear. The resulting translation is faithful for the supported model semantics, which follows from Lemma 3.16 (Lemma 3.16 of Strass, 2013).

5.1.2 Representing Bipolar Boolean Functions

While bipolarity has hitherto predominantly been defined and used in the context of ADFs (Brewka and Woltran, 2010), it is easy to define the concept for Boolean functions in general. Let A be a set of atoms and $f : 2^A \to \{\mathbf{t}, \mathbf{f}\}$ be a Boolean function. An atom $a \in A$ is *supporting* iff for all $M \subseteq A$, $f(M) = \mathbf{t}$ implies $f(M \cup \{a\}) = \mathbf{t}$; we then write $a \in sup(f)$. An atom $a \in A$ is *attacking* iff for all $M \subseteq A$, $f(M) = \mathbf{f}$ implies $f(M \cup \{a\}) = \mathbf{f}$; we then write $a \in att(f)$. A Boolean function $f : 2^A \to \{\mathbf{t}, \mathbf{f}\}$ is *semantically bipolar* iff each $a \in A$ is supporting or attacking or both. Throughout this chapter, we will sometimes take a Boolean function to be given by an interpretation set and then say that the set is bipolar.

We will now define bipolar propositional formulas for representing bipolar ADFs. This is important not only for our study, but also since (for three-valued semantics), bipolarity is the key to BADFs' low complexity in comparison to general ADFs (Chapter 4). Up to now, we usually assumed that to specify a bipolar ADF, in addition to statements, links and acceptance conditions, the user specifies for each link whether it is supporting or attacking. Here we introduce an arguably simpler way, where support and attack is represented (implicitly) in the syntax of the propositional formula encoding the acceptance function.

Formally, the *polarity* of the occurrence of an atom $a \in A$ in a formula is determined by the number of negations on the path from the root of the formula tree to the atom. The polarity is *positive* if the number is even and *negative* if the number is odd.

Definition 5.4. A propositional formula φ over A is *syntactically bipolar* if and only if no atom $a \in A$ occurs both positively and negatively in φ. ◇

Recall that we only use formulas over the basis $\{\wedge, \vee, \neg\}$ and thus there are no "hidden" negations, e.g. from material implication. For formulas in negation normal form (that is, where negation is only applied to atomic formulas), the polarities of the atoms can be read off the formula directly.

We will now address the question how to represent bipolar Boolean functions. Clearly all Boolean functions can be represented by propositional formulas; we modify this construction later and thus reproduce it here: for a Boolean function $f : 2^A \to \{\mathbf{t}, \mathbf{f}\}$, its associated formula is

$$\varphi_f = \bigvee_{M \subseteq A, f(M) = \mathbf{t}} \varphi_M \quad \text{with} \quad \varphi_M = \bigwedge_{a \in M} a \wedge \bigwedge_{a \in A \setminus M} \neg a \tag{5.1}$$

That is, each φ_M has exactly one model M, and φ_f enumerates those models.

So in particular, all bipolar Boolean functions can be represented by propositional formulas as well. However, this only guarantees us the existence of such representations but gives us no way to actually obtain them. Our first fundamental result shows how we can construct a syntactically bipolar propositional formula from a given semantically bipolar Boolean function. The converse is straightforward, and thus the two notions of bipolarity are closely related. For a formula φ, its associated Boolean function f_φ returns $\mathbf{t}$ if and only if it gets as input a model of φ.

Theorem 5.1. *Let A be a set of atoms.*

1. *For each syntactically bipolar formula φ over A, its Boolean function f_φ is semantically bipolar.*
2. *For each semantically bipolar Boolean function $f : 2^A \to \{\mathbf{t}, \mathbf{f}\}$, a syntactically bipolar formula ψ_f with $f_{\psi_f} = f$ is given by*

$$\psi_f = \bigvee_{\substack{M \subseteq A, \\ f(M) = \mathbf{t}}} \psi_M \quad \text{with} \quad \psi_M = \bigwedge_{\substack{a \in M, \\ a \notin att(f)}} a \wedge \bigwedge_{\substack{a \in A \setminus M, \\ a \notin sup(f)}} \neg a \tag{5.2}$$

Proof. 1. *Obvious: every atom occurring only positively is supporting, every atom occurring only negatively is attacking.*

2. *Let $f : 2^A \to \{\mathbf{t}, \mathbf{f}\}$ be semantically bipolar. Note first that by (5.2), for any $M \subseteq A$ we have $\models \varphi_M \to \psi_M$. It is easy to see that ψ_f is syntactically bipolar: Since f is semantically bipolar, each $a \in A$ is: (1) attacking and not supporting, then it occurs only negatively in ψ_f; or (2) supporting and not attacking, then it occurs only positively in ψ_f; or (3) supporting and attacking, then it does not occur in ψ_f. It remains to show that $f_{\psi_f} = f$; we show $\models \varphi_f \leftrightarrow \psi_f$.*

   $\models \varphi_f \to \psi_f$: *Let $v : A \to \{\mathbf{t}, \mathbf{f}\}$ with $v(\varphi_f) = \mathbf{t}$. Then there is an $M \subseteq A$ such that $f(M) = \mathbf{t}$ and $v(\varphi_M) = \mathbf{t}$. (Clearly $v = v_M$.) By $\models \varphi_M \to \psi_M$ we get $v(\psi_M) = \mathbf{t}$ and thus $v(\psi_f) = \mathbf{t}$.*

   $\models \psi_f \to \varphi_f$: *For each model v of ψ_f, there is an $M \subseteq A$ with $f(M) = \mathbf{t}$ such that $v(\psi_M) = \mathbf{t}$. To show that each model of ψ_f is a model of φ_f, we show that for all $M \subseteq A$ with $f(M) = \mathbf{t}$, each model v of ψ_M is a model of φ_f. Let $|A| = n$. Then each φ_M contains exactly n literals. For the corresponding ψ_M there is a $k \in \mathbb{N}$ with $0 \leq k \leq n$ such that ψ_M contains exactly $n - k$ literals. For two interpretations $v_1 : A \to \{\mathbf{t}, \mathbf{f}\}$ and $v_2 : A \to \{\mathbf{t}, \mathbf{f}\}$, define the* difference *between them as $\delta(v_1, v_2) = \{a \in A \mid v_1(a) \neq v_2(a)\}$. (Note that for $|A| = n$ we always have $|\delta(v_1, v_2)| \leq n$.) We will use induction on k to show the following: for each $M \subseteq A$ with $f(M) = \mathbf{t}$, each $v : A \to \{\mathbf{t}, \mathbf{f}\}$ with $v(\psi_M) = \mathbf{t}$ and $|\delta(v, v_M)| = k$ we find that $v(\varphi_f) = \mathbf{t}$. This covers all models v of ψ_f (since $|\delta(v, v_M)| \leq |A|$) and thus establishes the claim.*

   $k = 0$: *$\delta(v, v_M) = \emptyset$ implies $v = v_M$ whence $v(\varphi_f) = v_M(\varphi_f) = v_M(\varphi_M) = \mathbf{t}$ by definition of φ_M and φ_f.*

   $k \rightsquigarrow k + 1$: *Let $M \subseteq A$ with $f(M) = \mathbf{t}$, and $v : A \to \{\mathbf{t}, \mathbf{f}\}$ with $v(\psi_M) = \mathbf{t}$ and $|\delta(v, v_M)| = k + 1$. Since $k + 1 > 0$, there is some $a \in \delta(v, v_M)$, that is, an $a \in A$ with $v(a) \neq v_M(a)$.*

   (a) *a is supporting and not attacking. Then necessarily $v(a) = \mathbf{t}$. (If $v(a) = \mathbf{f}$, then $v_M(a) \neq v(a)$ implies $v_M(a) = \mathbf{t}$, that is, $a \in M$ whence $\{\psi_M\} \models a$ and $v(\psi_M) = \mathbf{f}$, contradiction.) Define the interpretation $w : A \to \{\mathbf{t}, \mathbf{f}\}$ such that $w(a) = \mathbf{f}$ and $w(c) = v(c)$ for $c \in A \setminus \{a\}$. Clearly $\delta(v, w) = \{a\}$ and $|\delta(w, v_M)| = k$. Hence the induction hypothesis applies to w and $w(\varphi_f) = \mathbf{t}$. Now $w(a) = \mathbf{f}$, $v(a) = \mathbf{t}$ and $w(\varphi_f) = \mathbf{t}$. Since a is supporting, also $v(\varphi_f) = \mathbf{t}$.*

   (b) *a is attacking and not supporting. Symmetric to the opposite case above.*

   (c) *a is both supporting and attacking. Define interpretation $w : A \to \{\mathbf{t}, \mathbf{f}\}$ such that $w(a) = v_M(a)$ and $w(c) = v(c)$ for $c \in A \setminus \{a\}$. It follows that $|\delta(w, v_M)| = k$, whence the induction hypothesis applies to w and $w(\varphi_f) = \mathbf{t}$. Since a is both supporting and attacking (thus redundant), we get that $v(\varphi_f) = w(\varphi_f) = \mathbf{t}$.* □

This result paves the way for analysing the succinctness of bipolar ADFs, since now we have a quite natural way of representing them.

5.2 Relative Expressiveness

We now analyse and compare the relative expressiveness of argumentation frameworks (AFs), (bipolar) abstract dialectical frameworks ((B)ADFs), normal and canonical logic programs (NLPs/CLPs) and propositional logic (PL). We first look at the different families of semantics – supported and stable models – in isolation and afterwards combine the results for the two semantics. For the formalisms $\mathcal{F} \in \{\text{ADF}, \text{NLP}, \text{CLP}\}$ that have both supported and stable semantics, we will indicate the semantics σ via a superscript as in Definition 5.1. For AFs we only consider the stable semantics, as this is (to date) the only semantics for AFs where all interpretations are guaranteed to map all arguments to either true (accepted) or false (rejected, i.e. attacked by an accepted argument). For propositional logic PL we consider the usual model semantics.

With the syntactic translations we reviewed in the previous section, we currently have the following expressiveness relationships. For the supported semantics,

$$\text{AF} \leq_e \text{BADF}^{su} \leq_e \text{ADF}^{su} \cong_e \text{NLP}^{su} \leq_e \text{CLP}^{su} \leq_e \text{PL}$$

and for the stable semantics,

$$\text{AF} \leq_e \text{NLP}^{st} <_e \text{PL} \cong_e \text{CLP}^{st} \text{ and } \text{AF} \leq_e \text{BADF}^{st} \leq_e \text{ADF}^{st} <_e \text{PL}$$

Note that $\text{NLP}^{st} <_e \text{PL}$ and $\text{ADF}^{st} <_e \text{PL}$ hold since sets of stable models of NLPs have an antichain property, in contrast to model sets of propositional logic.

For the succinctness relation, we have

$$\text{AF} \leq_s \text{BADF}^{su} \leq_s \text{ADF}^{su} \leq_s \text{PL} \text{ and } \text{NLP}^{su} \leq_s \text{ADF}^{su}$$

5.2.1 Supported Semantics

As depicted above, we know that expressiveness from AFs to propositional logic does not decrease. However, it is not yet clear if any of the relationships is strict. In what follows we will show that two of them are strict, working our way top-down from most to least expressive.

ADF vs. PL

We first show that ADFs can realise any set of models by showing how a given propositional formula can be used to construct an equivalent ADF of linear size.[4]

Theorem 5.2. *$\text{PL} \leq_e \text{ADF}^{su}$ and $\text{PL} \leq_s \text{ADF}^{su}$.*

Proof. Let ψ be a propositional formula over vocabulary A. Define the ADF D_ψ over A by setting, for all $a \in A$,

$$\varphi_a = a \leftrightarrow \psi = (a \wedge \psi) \vee (\neg a \wedge \neg \psi)$$

Thus $\|\varphi_a\| \in O(\|\psi\|)$, whence $\|D_\psi\| \in O(|A| \cdot \|\psi\|)$. It remains to show $su(D_\psi) = mod(\psi)$. Recall that for any ADF D over A, $su(D) = mod(\Phi_D)$ for $\Phi_D = \bigwedge_{a \in A} (a \leftrightarrow \varphi_a)$. Applying the definition of φ_a in D_ψ yields

$$\Phi_{D_\psi} = \bigwedge_{a \in A} (a \leftrightarrow (a \leftrightarrow \psi))$$

[4] If we consider the vocabulary A to be part of the input, the size increase is quadratic.

Now for any $a \in A$, the formula $(a \leftrightarrow (a \leftrightarrow \psi))$ is equivalent to ψ. (The proof is by case distinction on a.) Thus Φ_{D_ψ} is equivalent to $\bigwedge_{a \in A} \psi$, that is, to ψ, and it follows that $su(D_\psi) = mod(\Phi_{D_\psi}) = mod(\psi)$. □

For example, consider the vocabulary $A = \{a, b\}$ and the propositional formula $\psi = a \wedge b$. The canonical construction above yields ADF D_ψ with acceptance formulas $\varphi_a = a \leftrightarrow (a \wedge b)$ and $\varphi_b = b \leftrightarrow (a \wedge b)$. Now we have:

$$\varphi_a = a \leftrightarrow (a \wedge b) = (a \rightarrow (a \wedge b)) \wedge ((a \wedge b) \rightarrow a) \equiv \neg a \vee (a \wedge b) \equiv \neg a \vee b$$

Intuitively, $\varphi_a = \neg a \vee b$ expresses that a cannot be false, and is true if b is true. By a symmetrical argument, the acceptance formula of b is equivalent to $\neg b \vee a$. It is readily checked that $su(D_\psi) = \{\{a, b\}\}$ as desired. Since we know from Section 5.1.1 that the converse translation is also possible (in symbols ADF$^{su} \leq_s$ PL), we get the following.

Corollary 5.3. *PL $\cong_s$ ADF^{su}*

When the acceptance conditions are written as propositional formulas, the construction to realise $X \subseteq 2^A$ in the proof of Theorem 5.2 defines a space-efficient equivalent of

$$\varphi_a = \bigvee_{M \in X, a \in M} \varphi_M \vee \bigvee_{M \subseteq A, M \notin X, a \notin M} \varphi_M$$

as acceptance formula of a, where φ_M is as in Footnote 2.

ADF vs. NLP

Since ADFs under supported semantics can be faithfully translated into normal logic programs, which can be likewise further translated to propositional logic, we have the following.

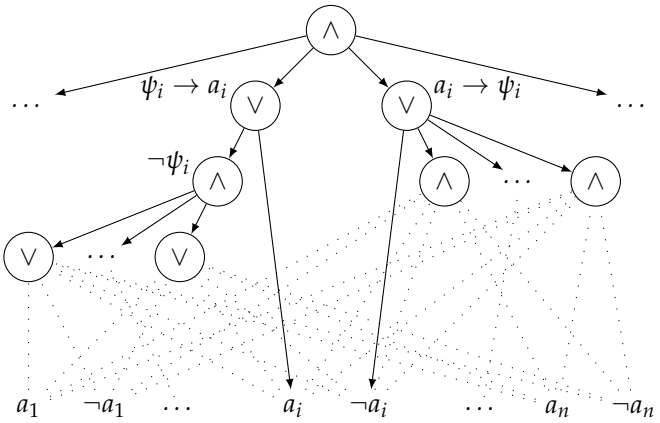
Corollary 5.4. *ADF^{su} $\cong_e$ NLP^{su} $\cong_e$ PL*

However, this does not extend to the succinctness relation, as logic programs stipulate a particular syntactic form that is essentially a fixed-depth circuit. More specifically, it is easy to see that any language that is polynomially expressible by normal logic programs under supported semantics is in AC^0. For the stable semantics of so-called canonical logic programs, this has recently been shown by Shen and Zhao (2014) (Proposition 2.1). The case we are interested in (supported semantics) works similarly, but we still present the proof for completeness. The main technical result towards proving that is a lemma showing how to turn a logic program into an equivalent Boolean circuit of a fixed depth.

Lemma 5.5. *For every normal logic program P, there exists a circuit C_P over the basis $\{\neg, \wedge, \vee\}$ such that:*

1. *C_P accepts all and only the supported models of P,*
2. *the size of C_P is linear the size of P,*
3. *C_P has depth 4.*

Proof. Let $A = \{a_1, \ldots, a_n\}$ be the vocabulary of P, and its Clark completion be $\Phi_P = \{a_i \leftrightarrow \psi_i \mid a_i \in A\}$ where the ψ_i are DNFs over literals from A. Clearly the circuit for Φ_P must compute $C_P = \bigwedge_{a_i \in A}(a_i \leftrightarrow \psi_i)$ where $a_i \leftrightarrow \psi_i$ can be replaced by $(\neg a_i \vee \psi_i) \wedge (a_i \vee \neg\psi_i)$ with $\neg\psi_i$ a CNF over literals from A. The construction can be depicted as follows, where the inner layers are shown for one i only, and dotted lines represent potential edges.



Now (1) follows since $su(P) = mod(\Phi_P)$ and C_P accepts all and only the models of Φ_P. For (2), if P contains $m = |P|$ rules, then $m \leq \|P\|$ and the total number of inner gates is bounded by $n(2m+3) \leq n(2 \cdot \|P\| + 3)$. (3) is clear. □

While the statement of Lemma 5.5 is actually much stronger and gives a *constant* upper bound of the resulting circuit depth for arbitrarily-sized logic programs, it readily follows that the set of polynomially logic-program expressible languages is a subset of the languages expressible by alternating Boolean circuits with unbounded fan-in and constant depth.

Proposition 5.6. *If L is polynomially expressible by normal logic programs under supported semantics, then $L \in \mathsf{AC}^0$.*

It follows immediately that normal logic programs cannot polynomially express the language Parity.[5] This is the supported-semantics counterpart of Theorem 3.1 in (Shen and Zhao, 2014).

Corollary 5.7. Parity *has no polynomial size normal logic program representation.*

Proof. By Proposition 5.6 and Parity $\notin \mathsf{AC}^0$ *(Jukna, 2012).* □

It follows that propositional logic is strictly more succinct than normal logic programs under supported semantics.

Corollary 5.8. *PL $\not\leq_s$ NLP^{su} and thus NLP^{su} $<_s$ PL.*

From our considerations since Theorem 5.2, it follows that if ψ has a "small" conjunctive normal form (a conjunction of clauses) and disjunctive normal form (disjunction of monomials) representation, then there is also a "small" normal logic program representation for $mod(\psi)$.

ADF vs. BADF

It is quite obvious that the canonical ADF constructed in Theorem 5.2 is not bipolar, since a as well as every atom mentioned by ψ occurs both positively and negatively in φ_a. This raises the question whether the construction can be adapted to *bipolar* ADFs.

It turns out that the subclass of bipolar ADFs is strictly less expressive. Towards the proof of this result we start out with a new concept: that of the conjugate of a model set with

[5] Logic programs under supported models are universally expressive, so they *can* express Parity, just not in polynomial size.

respect to an atom. This concept will be used to characterise ADF realisability and precisely captures the "if-and-only-if part" of ADFs' supported model semantics: From the translation of an ADF D into propositional logic (cf. Section 5.1.1) we can see that the result is basically a conjunction of equivalences: $\phi_D = \bigwedge_{a \in A}(a \leftrightarrow \varphi_a)$. While the conjunction part will be captured by set intersection, the conjugate will capture the equivalence part.

Definition 5.5. Let A be a vocabulary, $X \subseteq 2^A$ and $a \in A$. The *a-conjugate of* X is the set

$$\langle a \rangle(X) = \{M \subseteq A \mid M \in X, a \in M\} \cup \{M \subseteq A \mid M \notin X, a \notin M\} \qquad \diamond$$

Alternatively, we could write $\langle a \rangle(X) = \{M \subseteq A \mid M \in X \text{ iff } a \in M\}$. Intuitively, $\langle a \rangle(X)$ contains all interpretations M where containment of a in M coincides exactly with containment of M in X. Formulated in terms of propositional formulas, if X is the model set of formula φ over A, then $\langle a \rangle(X)$ is the model set of formula $a \leftrightarrow \varphi$. Note that the vocabulary A is implicit in the conjugate function.

Example 5.1. Consider the vocabulary $A = \{a, b\}$. The functions $\langle a \rangle(\cdot)$ and $\langle b \rangle(\cdot)$ operate on the set 2^{2^A} of interpretation sets over A and are shown in Table 5.1.

φ	$\langle a \rangle(\varphi)$	$\langle b \rangle(\varphi)$
$\bot$	$\neg a$	$\neg b$
$\neg a \wedge \neg b$	$\neg a \wedge b$	$a \wedge \neg b$
$a \wedge \neg b$	$\neg a \vee \neg b$	$\neg a \wedge \neg b$
$\neg a \wedge b$	$\neg a \wedge \neg b$	$\neg a \vee \neg b$
$a \wedge b$	$a \rightarrow b$	$b \rightarrow a$
a	$\top$	$a \leftrightarrow b$
b	$a \leftrightarrow b$	$\top$
$\neg a$	$\bot$	$a \veebar b$
$\neg b$	$a \veebar b$	$\bot$
$a \veebar b$	$\neg b$	$\neg a$
$a \leftrightarrow b$	b	a
$a \vee b$	$b \rightarrow a$	$a \rightarrow b$
$\neg a \vee \neg b$	$a \wedge \neg b$	$\neg a \wedge b$
$a \rightarrow b$	$a \wedge b$	$a \vee b$
$b \rightarrow a$	$a \vee b$	$a \wedge b$
$\top$	a	b

Table 5.1: Conjugation functions for $A = \{a, b\}$. Interpretation sets are represented using formulas over A, and connective "$\veebar$" denotes exclusive disjunction XOR.

For two-valued ADF semantics, this conjugation function plays an essential semantical role, since it provides the "bridge" between models of the acceptance functions and models of the ADF. But it is also interesting in itself: We first show some properties of the conjugation function associated to an atom, since some of them will be used in the proof later on. First of all, it is an involution, that is, its own inverse (and thus in particular a bijection). Next, it is compatible with the complement operation (logical negation on the formula level). Finally, it also preserves the evenness of the cardinality of the input set.

Proposition 5.9. *Let A be a vocabulary, $X \subseteq 2^A$ and $a \in A$.*

1. $\langle a\rangle(\langle a\rangle(X)) = X$. *(involution)*
2. $2^A \setminus \langle a\rangle(X) = \langle a\rangle\big(2^A \setminus X\big)$. *(compatible with negation)*
3. $|X|$ *is even iff* $|\langle a\rangle(X)|$ *is even.* *(preserves evenness)*

Proof. *Let* $|A| = n$, $X \subseteq 2^A$ *and* $a \in A$.

1. *Let* $M \subseteq A$. *We have*

$$\begin{aligned} M \in \langle a\rangle(\langle a\rangle(X)) \text{ iff } & M \in \langle a\rangle(X) \leftrightarrow a \in M \\ \text{iff } & (M \in X \leftrightarrow a \in M) \leftrightarrow a \in M \\ \text{iff } & M \in X \leftrightarrow (a \in M \leftrightarrow a \in M) \\ \text{iff } & M \in X \end{aligned}$$

2. *Denote*

$$\begin{aligned} S_{\in,\in} &= \{M \subseteq A \mid M \in X, a \in M\} \\ S_{\in,\notin} &= \{M \subseteq A \mid M \in X, a \notin M\} \\ S_{\notin,\in} &= \{M \subseteq A \mid M \notin X, a \in M\} \\ S_{\notin,\notin} &= \{M \subseteq A \mid M \notin X, a \notin M\} \end{aligned}$$

and observe that

$$\begin{aligned} 2^A &= S_{\in,\in} \dot\cup S_{\in,\notin} \dot\cup S_{\notin,\in} \dot\cup S_{\notin,\notin} \\ X &= S_{\in,\in} \dot\cup S_{\in,\notin} \\ \langle a\rangle(X) &= S_{\in,\in} \dot\cup S_{\notin,\notin} \end{aligned}$$

where $\dot\cup$ *denotes disjoint union. Now*

$$\begin{aligned} 2^A \setminus \langle a\rangle(X) &= 2^A \setminus (S_{\in,\in} \dot\cup S_{\notin,\notin}) \\ &= S_{\in,\notin} \dot\cup S_{\notin,\in} \\ &= \{M \subseteq A \mid M \in X, a \notin M\} \cup \{M \subseteq A \mid M \notin X, a \in M\} \\ &= \Big\{M \subseteq A \;\Big|\; M \notin 2^A \setminus X, a \notin M\Big\} \cup \Big\{M \subseteq A \;\Big|\; M \in 2^A \setminus X, a \in M\Big\} \\ &= \langle a\rangle\Big(2^A \setminus X\Big) \end{aligned}$$

3. *We show that* $|X| + |\langle a\rangle(X)|$ *is even. Firstly,*

$$S_{\in,\notin} \dot\cup S_{\notin,\notin} = \{M \subseteq A \mid a \notin M\} = 2^{A\setminus\{a\}}$$

whence $\big|S_{\in,\notin}\big| + \big|S_{\notin,\notin}\big| = 2^{n-1}$. *Thus*

$$\begin{aligned} |X| + |\langle a\rangle(X)| &= \big(|S_{\in,\in}| + \big|S_{\in,\notin}\big|\big) + \big(|S_{\in,\in}| + \big|S_{\notin,\notin}\big|\big) \\ &= 2 \cdot |S_{\in,\in}| + \big|S_{\in,\notin}\big| + \big|S_{\notin,\notin}\big| \\ &= 2 \cdot |S_{\in,\in}| + 2^{n-1} \end{aligned}$$

is even. □

For our current purpose of characterising the expressiveness of bipolar ADFs, we now use the concept of conjugation to make ADF realisability for the model semantics slightly more accessible. We show that each ADF realisation of a model set X over an n-element vocabulary A can equivalently be characterised by an n-tuple $(Y_1, \ldots, Y_n)$ of supersets of X whose intersection is exactly X. The crux of the proof of this result is how the acceptance conditions of the realising ADF and the Y_i are related through the conjugation function.

Proposition 5.10. *Let $A = \{a_1, \ldots, a_n\}$ be a vocabulary and $X \subseteq 2^A$ be a set of interpretations. Denote an ADF over A by the sequence $(\varphi_1, \ldots, \varphi_n)$ of its acceptance formulas (for each $i \in \{1, \ldots, n\}$, formula φ_i over A is the acceptance formula of a_i), and further define*

$$C_X = \{(mod(\varphi_1), \ldots, mod(\varphi_n)) \mid su(\varphi_1, \ldots, \varphi_n) = X\}$$
$$Y_X = \left\{ (Y_1, \ldots, Y_n) \;\middle|\; Y_1, \ldots, Y_n \subseteq 2^A, \left(\bigcap_{i=1}^{n} Y_i\right) = X \right\}$$

The sets C_X and Y_X are in one-to-one correspondence; in particular $|C_X| = |Y_X|$.

Proof. We provide a bijection between C_X and Y_X. Consider the function

$$f : \left(2^{2^A}\right)^n \to \left(2^{2^A}\right)^n \quad \textit{with} \quad (B_1, \ldots, B_n) \mapsto (\langle a_1 \rangle (B_1), \ldots, \langle a_n \rangle (B_n))$$

which is an involution by Proposition 5.9. Using the results of Section 5.1.1, we get that

$$\begin{aligned}
(mod(\varphi_1), \ldots, mod(\varphi_n)) \in C_X \;\; \textit{iff} \;\; & su(\varphi_1, \ldots, \varphi_n) = X \\
\textit{iff} \;\; & mod\left(\bigwedge_{1 \le i \le n} (a_i \leftrightarrow \varphi_i)\right) = X \\
\textit{iff} \;\; & \bigcap_{1 \le i \le n} mod(a_i \leftrightarrow \varphi_i) = X \\
\textit{iff} \;\; & \bigcap_{1 \le i \le n} \langle a_i \rangle (mod(\varphi_i)) = X \\
\textit{iff} \;\; & (\langle a_1 \rangle (mod(\varphi_1)), \ldots, \langle a_n \rangle (mod(\varphi_n))) \in Y_X \\
\textit{iff} \;\; & f(mod(\varphi_1), \ldots, mod(\varphi_n)) \in Y_X
\end{aligned}$$

Thus $f(C_X) = Y_X$ whence $f(Y_X) = f(f(C_X)) = C_X$ and $f|_{C_X} : C_X \to Y_X$ is bijective. □

This one-to-one correspondence is important since we will later analyse the precise number of realisations of given model sets. Furthermore, this result shows the role of the conjugation function for characterising two-valued model realisability for general ADFs. We can now adapt this characterisation result to the case of bipolar ADFs. More precisely, we give several necessary and sufficient conditions when a given model set is bipolarly realisable. With this characterisation in hand, we can later show that a specific interpretation set fails the necessary conditions and thus cannot be the model set of any BADF. Below, we denote the set of all supersets of a set X of interpretation sets over A by $X^{\uparrow} = \{Y \subseteq 2^A \mid X \subseteq Y\}$.

Proposition 5.11. *Let $A = \{a_1, \ldots, a_n\}$ be a vocabulary and $X \subseteq 2^A$ be a set of interpretations. The following are equivalent:*

1. *X is bipolarly realisable.*

2. *there exist* $Y_1, \ldots, Y_n \in X^{\uparrow}$ *such that:*
   (a) $(\bigcap_{i=1}^{n} Y_i) = X$, *and*
   (b) *for each* $1 \leq i \leq n$, *the set* $\langle a_i \rangle(Y_i)$ *is bipolar.*
3. *there exist* $Y_1, \ldots, Y_n \in X^{\uparrow}$ *such that*
   (a) $(\bigcap_{i=1}^{n} Y_i) = X$, *and*
   (b) *for each* $1 \leq i, j \leq n$, *at least one of:*
      - *for all* $M \subseteq A$, $(M \in Y_i \leftrightarrow a_i \in M) \rightarrow (M \cup \{a_j\} \in Y_i \leftrightarrow a_i \in M \cup \{a_j\})$; *or*
      - *for all* $N \subseteq A$, $(N \in Y_i \mathbin{\dot{\vee}} a_i \in N) \rightarrow (N \cup \{a_j\} \in Y_i \mathbin{\dot{\vee}} a_i \in N \cup \{a_j\})$.

Proof. (*Item* 1) $\Longrightarrow$ (2): *If* X *is bipolarly realisable, then there exists a bipolar ADF* $D = (A, L, C)$ *with* $su(D) = X$. *In particular, there exist bipolar Boolean functions* $C_1, \ldots, C_n$ *such that* $M \in X$ *if and only if for all* $1 \leq i \leq n$ *we find* $a_i \in M$ *iff* $C_i(M) = \mathbf{t}$. *For each* $1 \leq i \leq n$ *define* $Y_i = \langle a_i \rangle(C_i)$. *By assumption,* $\langle a_i \rangle(Y_i) = \langle a_i \rangle(\langle a_i \rangle(C_i)) = C_i$ *is bipolar; furthermore* $(\bigcap_{i=1}^{n} Y_i) = X$ *follows from the above.*

(2) $\Longrightarrow$ (3): *Let* $i \in \{1, \ldots, n\}$ *and assume that* $\langle a_i \rangle(Y_i)$ *is bipolar. This means that for all* $a_j \in A$, *we find that* a_j *is supporting or attacking (or both) in* $\langle a_i \rangle(Y_i)$. *Now* a_j *is supporting in* $\langle a_i \rangle(Y_i)$ *iff for all* $M \subseteq A$ *we find:*

$$M \in \langle a_i \rangle(Y_i) \rightarrow M \cup \{a_j\} \in \langle a_i \rangle(Y_i)\text{, that is,}$$
$$(M \in Y_i \leftrightarrow a_i \in M) \rightarrow (M \cup \{a_j\} \in Y_i \leftrightarrow a_i \in M \cup \{a_j\})$$

Similarly, a_j *is attacking in* $\langle a_i \rangle(Y_i)$ *iff for all* $N \subseteq A$ *we find:*

$$N \notin \langle a_i \rangle(Y_i) \rightarrow N \cup \{a_j\} \notin \langle a_i \rangle(Y_i)\text{, that is,}$$
$$\neg(N \in Y_i \leftrightarrow a_i \in N) \rightarrow \neg(N \cup \{a_j\} \in Y_i \leftrightarrow a_i \in N \cup \{a_j\})$$

Thus for all $a_j \in A$, *we find that at least one of the following:*

- *for all* $M \subseteq A$, $(M \in Y_i \leftrightarrow a_i \in M) \rightarrow (M \cup \{a_j\} \in Y_i \leftrightarrow a_i \in M \cup \{a_j\})$; *or*
- *for all* $N \subseteq A$, $(N \in Y_i \mathbin{\dot{\vee}} a_i \in N) \rightarrow (N \cup \{a_j\} \in Y_i \mathbin{\dot{\vee}} a_i \in N \cup \{a_j\})$.

(3) $\Longrightarrow$ (1): *We construct an ADF* $D = (A, L, C)$ *as follows: for each* $i \in \{1, \ldots, n\}$ *we define* $C_i = \langle a_i \rangle(Y_i)$ *and finally set* $L = A \times A$. *Each* C_i *is bipolar by the equivalences established in the previous proof item, and* $su(D) = X$ *follows from the fact that* $\langle a_i \rangle(C_i) = \langle a_i \rangle(\langle a_i \rangle(Y_i)) = Y_i$ *and the presumption* $(\bigcap_{i=1}^{n} Y_i) = X$. □

We now apply this characterisation result to show that there is an interpretation set over three atoms that cannot be realised by bipolar ADFs under the model semantics. This is the smallest example in terms of the number of atoms (actually, one of the two smallest examples) – all interpretation sets over a binary vocabulary are bipolarly realisable.

Proposition 5.12. *For vocabulary* $A_3 = \{1, 2, 3\}$, *there is no bipolar ADF that realises* $X = \textsc{Even}_3 = \{\emptyset, \{1, 2\}, \{1, 3\}, \{2, 3\}\}$.

Proof. Assume to the contrary that X is bipolarly realisable. Then there exist $Y_1, Y_2, Y_3 \in X^{\uparrow}$ from Proposition 5.11. There are $2^{|2^A|-|X|} = 2^{8-4} = 2^4 = 16$ candidates for each Y_i, that is, every Y_i must be of the form $X \dot\cup Z$ with

$$Z \subseteq \{\{1\},\{2\},\{3\},\{1,2,3\}\} = 2^A \setminus X$$

For eleven out of those sixteen model set candidates for each Y_i, the set $\langle i\rangle(Y_i)$ is not bipolar. To show that a model set $\langle i\rangle(Y_i)$ is not bipolar, we provide a statement $j \in A_3$ that is neither supporting nor attacking; we say that such a statement is dependent.

1. *For $Y_1 = X$, we get $\langle 1\rangle(Y_1) = \{\{1,2\},\{1,3\},\{2\},\{3\}\}$, which is not bipolar since statement 2 is dependent: If 2 was supporting, then $\{3\} \in \langle 1\rangle(Y_1)$ would imply $\{2,3\} \in \langle 1\rangle(Y_1)$; if 2 was attacking, then $\emptyset \notin \langle 1\rangle(Y_1)$ would imply $\{2\} \notin \langle 1\rangle(Y_1)$. For the remaining cases, the justifications for a specific statement being dependent are equally easy to read off the model set; for brevity we just indicate the statements.*
2. *For $Y_1 = X \cup \{\{1\}\}$, we get $\langle 1\rangle(Y_1) = \{\{1,2\},\{1,3\},\{1\},\{2\},\{3\}\}$, which is not bipolar since statement 2 is dependent.*
3. *For $Y_1 = X \cup \{\{2\}\}$, we get $\langle 1\rangle(Y_1) = \{\{1,2\},\{1,3\},\{3\}\}$, which is not bipolar since statement 2 is dependent.*
4. *The case $Y_1 = X \cup \{\{3\}\}$ is symmetric to the previous one: we get the model set $\langle 1\rangle(Y_1) = \{\{1,2\},\{1,3\},\{2\}\}$, which is not bipolar since statement 3 is dependent.*
5. *For $Y_1 = X \cup \{\{1,2,3\}\}$, we get $\langle 1\rangle(Y_1) = \{\{1,2,3\},\{1,2\},\{1,3\},\{2\},\{3\}\}$, which is not bipolar since statement 2 is dependent.*
6. *For $Y_1 = X \cup \{\{1\},\{2\}\}$, we get $\langle 1\rangle(Y_1) = \{\{1,2\},\{1,3\},\{1\},\{3\}\}$, which is not bipolar since statement 3 is dependent.*
7. *The case $Y_1 = X \cup \{\{1\},\{3\}\}$ is again symmetric to the previous one.*
8. *For $Y_1 = X \cup \{\{2\},\{3\}\}$, we get $\langle 1\rangle(Y_1) = \{\{1,2\},\{1,3\}\}$, which is not bipolar since statement 2 is dependent.*
9. *For $Y_1 = X \cup \{\{1\},\{1,2,3\}\}$, we get $\langle 1\rangle(Y_1) = \{\{1,2,3\},\{1,2\},\{1,3\},\{1\},\{2\},\{3\}\}$, which is not bipolar since statement 2 is dependent.*
10. *For $Y_1 = X \cup \{\{2\},\{1,2,3\}\}$, we get $\langle 1\rangle(Y_1) = \{\{1,2,3\},\{1,2\},\{1,3\},\{3\}\}$, which is not bipolar since statement 2 is dependent.*
11. *$Y_1 = X \cup \{\{3\},\{1,2,3\}\}$ is again symmetric to the previous case.*

There remains a set C of five candidates (due to symmetry they are the same for each i):

$$\begin{aligned} C = \{&X \dot\cup \{\{1\},\{2\},\{3\}\},\\ &X \dot\cup \{\{1\},\{2\},\{1,2,3\}\},\\ &X \dot\cup \{\{1\},\{3\},\{1,2,3\}\},\\ &X \dot\cup \{\{2\},\{3\},\{1,2,3\}\},\\ &X \dot\cup \{\{1\},\{2\},\{3\},\{1,2,3\}\}\} \end{aligned}$$

Basically, the candidates are those where at least three out of the four interpretations in $D = \{\{1\},\{2\},\{3\},\{1,2,3\}\}$ are contained in addition to those already in X. Now clearly by the

assumption that the Y_i realise X we have $Y_1, Y_2, Y_3 \in C$. But then there is some $M \in D$ with $M \in Y_i$ for all $1 \leq i \leq 3$ and thus $M \in \left(\bigcap_{i=1}^{3} Y_i\right) = X$. However, $D \cap X = \emptyset$. Contradiction. Thus such Y_i do not exist and X is not bipolarly realisable. □

As the only other interpretation set over A_3 that is not bipolarly realisable, we found the complement of $\textsc{Even}_3$ above, the $\textsc{Parity}$ language over three atoms.

Proposition 5.13. *For vocabulary $A_3 = \{1,2,3\}$, there is no bipolar ADF that realises $\textsc{Parity}_3 = \{\{1\},\{2\},\{3\},\{1,2,3\}\}$.*

Together with the straightforward statement of fact that $\textsc{Even}_3$ can be realised by a non-bipolar ADF, Proposition 5.12 leads to the next result.

Theorem 5.14. *$BADF^{su} <_e ADF^{su}$*

Proof. Model set $\textsc{Even}_3$ from Proposition 5.12 is realisable under model semantics by ADF $D_{\textsc{Even}_3}$ with acceptance conditions

$$\varphi_1 = (2 \dot{\vee} 3), \quad \varphi_2 = (1 \dot{\vee} 3), \quad \varphi_3 = (1 \dot{\vee} 2)$$

However, there is no bipolar ADF realising $\textsc{Even}_3$, as is witnessed by Proposition 5.12. □

Another consequence of our characterisation of two-valued model realisability in Proposition 5.10 is that we can get a precise number of distinct realisations of a given model set. This is significant in that it further illustrates the rather intricate difficulty underlying bipolar non-realisability: we cannot necessarily use the model set $\textsc{Even}_3$ above to determine a *single* reason for bipolar non-realisability, that is, a *single* link (b, a) that is neither supporting nor attacking in *all* realisations. Rather, the culprit(s) might be different in each realisation, and to show bipolar non-realisability, we have to prove that *for all* realisations, there necessarily *exists some* reason for non-bipolarity. And the number of different ADF realisations of a given model set X can be considerable.[6]

Proposition 5.15. *Let A be a vocabulary with $|A| = n$, and $X \subseteq 2^A$ an interpretation set with $\left|2^A \setminus X\right| = m$. The number of distinct ADFs D with $su(D) = X$ is*

$$r(n,m) = (2^n - 1)^m$$

Proof. According to Proposition 5.10, each realisation of X can be characterised by a tuple $(Y_1, \ldots, Y_n) \in \left(X^{\uparrow}\right)^n$ with $X = \bigcap_{i=1}^{n} Y_i$. Since $\left|X^{\uparrow}\right| = 2^m$, there are $(2^m)^n$ such tuples. However, towards $r(n,m)$, this wrongly counts all tuples $(Y_1, \ldots, Y_n)$ with $(\bigcap_{i=1}^{n} Y_i) \supsetneq X$, that is, $|(\bigcap_{i=1}^{n} Y_i) \setminus X| > 0$ (at least once); it remains to subtract them. For any $i \in \{1, \ldots, n\}$, we can overestimate the number of tuples $(Y_1, \ldots, Y_n) \in \left(X^{\uparrow}\right)^n$ such that $|(\bigcap_{i=1}^{n} Y_i) \setminus X| \geq i$ by the expression

$$q(n,m,i) = \binom{m}{i} \left(2^{m-i}\right)^n \tag{5.3}$$

This is seen as follows: Let $I \subseteq \left(2^A \setminus X\right)$ be a fixed i-element set. (Intuitively, the interpretation-set $X \cup I$ contains exactly i interpretations too many.) There are $\binom{m}{i}$ such sets. For each such I, we have $\left|I^{\uparrow}\right| = 2^{m-i}$. Thus there are $\left(2^{m-i}\right)^n$ possible ways to choose n elements (the $Y_1, \ldots, Y_n$) out of $I^{\uparrow}$. No matter how the Y_j are chosen, their intersection contains I and thus has at least i elements too many. However, all sets that have at least $i+1$ elements too many are counted twice and have to be subtracted.

[6] When counting ADFs over A, we do not take into account different link relations, but take $L = A \times A$ and only count different acceptance functions, through which redundant links can be modelled.

If we subtract $q(n,m,i+1)$, then we have not counted the sets that have at least $i+2$ elements too many and have to add $q(n,m,i+2)$, etc. Hence by the inclusion-exclusion principle, the number of tuples $(Y_1,\dots,Y_n) \in \left(X^{\uparrow}\right)^n$ with $\bigcap_{i=1}^n Y_i = X$ is given by

$$\begin{aligned}
r(n,m) &= q(n,m,0) - q(n,m,1) + q(n,m,2) - \ldots \pm q(n,m,m) \\
&= \sum_{i=0}^{m} (-1)^i q(n,m,i) \\
&= \sum_{i=0}^{m} (-1)^i \binom{m}{i} \left(2^{m-i}\right)^n && \textit{(by (5.3) above)} \\
&= \sum_{i=0}^{m} \binom{m}{i} (2^n)^{m-i} (-1)^i && \textit{(reordering factors)} \\
&= (2^n - 1)^m && \textit{(binomial theorem)} \qquad \square
\end{aligned}$$

So the main contributing factor is the number m of interpretations that are excluded from the desired model set X. For Proposition 5.12, for instance, there are $(2^3-1)^4 = 7^4 = 2401$ ADFs with the model set $\textsc{Even}_3$. According to Theorem 5.14, none of them is bipolar. Obviously, the maximal number of realisations is achieved by $X = \emptyset$ whence $r(n,2^n) = (2^n-1)^{2^n}$. On the other hand, the model set $X = 2^A$ has exactly one realisation, $r(n,0) = 1$. Note that the number of (syntactically distinct) realisations for the other universally expressive formalisms, logic programs and propositional logic, is unbounded in general since we can add an arbitrary number of tautologies.

We finally show a reduction of the problem of bipolar realisability to propositional satisfiability. This approaches the problem from another angle (a possible implementation deciding bipolar realisability using a SAT solver), and provides the proof of Theorem 3 by Strass (2015b), which was not contained in that work.

For a given vocabulary A and set $X \subseteq 2^A$ be a set of interpretations, it is our aim to construct a propositional formula ϕ_X that is satisfiable if and only if X is bipolarly realisable. The propositional signature we use is the following: For each $a \in A$ and $M \subseteq A$, there is a propositional variable p_a^M that expresses whether $C_a(M) = \mathbf{t}$. This allows to encode all possible acceptance conditions for the statements in A. To enforce bipolarity, we use additional variables to model supporting and attacking links: for all $a,b \in A$, there is a variable $p_{sup}^{a,b}$ saying that a supports b, and a variable $p_{att}^{a,b}$ saying that a attacks b. So the vocabulary of ϕ_X is given by

$$P = \left\{ p_a^M, p_{sup}^{a,b}, p_{att}^{a,b} \;\middle|\; M \subseteq A, a \in A, b \in A \right\}$$

To guarantee the desired set of models, we constrain the acceptance conditions as dictated by X: For any desired set M and statement a, the containment of a in M must correspond exactly to whether $C_a(M) = \mathbf{t}$; this is encoded in $\phi_X^{\in}$. Conversely, for any undesired set M and statement a, there must not be any such correspondence, which $\phi_X^{\notin}$ expresses. To enforce bipolarity, we state that each link must be supporting or attacking. To model the meaning of support and attack, we encode all ground instances of their definitions.

Definition 5.6. Let A be a vocabulary and $X \subseteq 2^A$ be a set of interpretations. Define the

following propositional formulas:

$$\phi_X^{\text{BADF}} = \phi_X^{\in} \wedge \phi_X^{\notin} \wedge \phi_{bipolar}$$

$$\phi_X^{\in} = \bigwedge_{M \in X} \left(\bigwedge_{a \in M} p_a^M \wedge \bigwedge_{a \in A \setminus M} \neg p_a^M \right)$$

$$\phi_X^{\notin} = \bigwedge_{M \subseteq A, M \notin X} \left(\bigvee_{a \in M} \neg p_a^M \vee \bigvee_{a \in A \setminus M} p_a^M \right)$$

$$\phi_{bipolar} = \bigwedge_{a,b \in A} \left(\left(p_{sup}^{a,b} \vee p_{att}^{a,b} \right) \wedge \phi_{sup}^{a,b} \wedge \phi_{att}^{a,b} \right)$$

$$\phi_{sup}^{a,b} = p_{sup}^{a,b} \rightarrow \bigwedge_{M \subseteq A} \left(p_b^M \rightarrow p_b^{M \cup \{a\}} \right) \qquad (a, b \in A)$$

$$\phi_{att}^{a,b} = p_{att}^{a,b} \rightarrow \bigwedge_{M \subseteq A} \left(p_b^{M \cup \{a\}} \rightarrow p_b^M \right) \qquad (a, b \in A) \qquad \diamond$$

The corresponding result shows the reduction to be correct.

Theorem 5.16. *Let A be a vocabulary and $X \subseteq 2^A$ be a set of interpretations. X is bipolarly realisable if and only if ϕ_X^{BADF} is satisfiable.*

Proof. "if": *Let $I \subseteq P$ be a model for ϕ_X. For each $a \in A$, we define an acceptance condition as follows: for $M \subseteq A$, set $C_a(M) = \mathbf{t}$ iff $p_a^M \in I$. It is easy to see that $\phi_{bipolar}$ guarantees that these acceptance conditions are all bipolar. The ADF is now given by $D_X^{su} = (A, A \times A, C)$. It remains to show that any $M \subseteq A$ is a model of D_X^{su} if and only if $M \in X$.*

"if": *Let $M \in X$. We have to show that M is a model of D_X^{su}. Consider any $a \in A$.*

1. *$a \in M$. Since I is a model of $\phi_X^{\in}$, we have $p_a^M \in I$ and thus by definition $C_a(M) = \mathbf{t}$.*
2. *$a \in A \setminus M$. Since I is a model of $\phi_X^{\in}$, we have $p_a^M \notin I$ and thus by definition $C_a(M) = \mathbf{f}$.*

"only if": *Let $M \notin X$. Since I is a model of $\phi_X^{\notin}$, there is an $a \in M$ such that $C_a(M) = \mathbf{f}$ or an $a \notin M$ such that $C_a(M) = \mathbf{t}$. In any case, M is not a model of D_X^{su}.*

"only if": *Let D be a bipolar ADF with $su(D) = X$. We use D to define a model I for ϕ_X. First, for $M \subseteq A$ and $a \in A$, set $p_a^M \in I$ iff $C_a(M) = \mathbf{t}$. Since D is bipolar, each link is supporting or attacking and for all $a, b \in A$ we can find a valuation for $p_{sup}^{a,b}$ and $p_{att}^{a,b}$. It remains to show that I is a model for ϕ_X.*

1. *I is a model for $\phi_X^{\in}$: Since D realises X, each $M \in X$ is a model of D and thus for all $a \in A$ we have $C_a(M) = \mathbf{t}$ iff $a \in M$.*
2. *I is a model for $\phi_X^{\notin}$: Since D realises X, each $M \subseteq A$ with $M \notin X$ is not a model of D. Thus for each such M, there is an $a \in A$ witnessing that M is not a model of D: (1) $a \in M$ and $C_a(M) = \mathbf{f}$, or (2) $a \notin M$ and $C_a(M) = \mathbf{t}$.*
3. *I is a model for $\phi_{bipolar}$: straightforward since D is bipolar by assumption.* $\square$

Remarkably, the decision procedure does not only give an answer, but in the case of a positive answer we can read off the BADF realisation from the satisfying evaluation of the constructed formula. We illustrate the construction with an example seen earlier.

Example 5.2. Consider $A_3 = \{1,2,3\}$ and the model set $\textsc{Even}_3 = \{\emptyset, \{1,2\}, \{1,3\}, \{2,3\}\}$. The construction of Theorem 5.16 yields these formulas:

$$\begin{aligned}\phi^{\in}_{\textsc{Even}_3} = {} & \neg p_1^{\emptyset} \wedge \neg p_2^{\emptyset} \wedge \neg p_3^{\emptyset} \wedge \\ & p_1^{\{1,2\}} \wedge p_2^{\{1,2\}} \wedge \neg p_3^{\{1,2\}} \wedge \\ & p_1^{\{1,3\}} \wedge \neg p_2^{\{1,3\}} \wedge p_3^{\{1,3\}} \wedge \\ & \neg p_1^{\{2,3\}} \wedge p_2^{\{2,3\}} \wedge p_3^{\{2,3\}}\end{aligned} \qquad \begin{aligned}\phi^{\notin}_{\textsc{Even}_3} = {} & \left(\neg p_1^{\{1\}} \vee p_2^{\{1\}} \vee p_3^{\{1\}}\right) \wedge \\ & \left(p_1^{\{2\}} \vee \neg p_2^{\{2\}} \vee p_3^{\{2\}}\right) \wedge \\ & \left(p_1^{\{3\}} \vee p_2^{\{3\}} \vee \neg p_3^{\{3\}}\right) \wedge \\ & \left(\neg p_1^{\{1,2,3\}} \vee \neg p_2^{\{1,2,3\}} \vee \neg p_3^{\{1,2,3\}}\right)\end{aligned}$$

The remaining formulas about bipolarity are independent of $\textsc{Even}_3$, we do not show them here. We have implemented the translation of the proof of Theorem 5.16 and used the solver clasp (Gebser, Kaminski, Kaufmann, Ostrowski, Schaub, and Schneider, 2011) to verify that $\phi_{\textsc{Even}_3}$ is unsatisfiable. ◇

BADF vs. NLP

Earlier, we used the language Parity to show that propositional logic is (and thus by PL $\cong_s$ ADF^{su} general ADFs are) exponentially more succinct than normal logic programs (under supported models). However, for bipolar ADFs, by Proposition 5.13 there is no BADF D over $A_3 = \{1,2,3\}$ with model set $su(D) = \textsc{Parity}_3 = \{\{1\}, \{2\}, \{3\}, \{1,2,3\}\}$, that is, BADFs cannot even express Parity. Fortunately, the Majority language does the trick in this case.

Theorem 5.17. *BADF^{su} $\not\leq_s$ NLP^{su}*

Proof. We show that the language Majority *can be polynomially expressed by BADF^{su}, but not by NLP^{su}. The latter fact follows from* Majority $\notin \mathsf{AC}^0$ *(Jukna, 2012) and Proposition 5.6. We show the first part by constructing a series of BADFs D_n over $A_n = \{a_1, \ldots, a_n\}$ ($n \in \mathbb{N}$, $n \geq 1$) such that $su(D_n) = \textsc{Majority}_n$. We use results of (Friedman, 1986; Boppana, 1986), who show that for all positive $n \in \mathbb{N}$ and $k \leq n$, the language $\textsc{Threshold}_{n,k}$ has negation-free propositional formulas $\Phi^{\textsc{Threshold}}_{n,k}$ of polynomial size s, where we use the bound of Boppana, $s \in O\left(k^{4.27} n \log n\right)$. Define D_1 by $\varphi_{a_1} = \top$, and for $n \geq 2$ set $k = \lceil \frac{n}{2} \rceil$ and for $1 \leq i \leq n$,*

$$\varphi_{a_i} = a_i \vee \neg \Phi^{\textsc{Threshold}}_{n-1,k}(a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n)$$

Intuitively, the formula φ_{a_i} checks whether the remaining variables could achieve a majority without a_i. If so, then a_i can be set arbitrarily; otherwise, a_i must be set to true. Clearly the Boolean function computed by φ_{a_i} is bipolar, since a_i is supporting and all other parents are attacking. For the size of D_n, we observe that

$$\|D_n\| \in O\left(n \left\| \Phi^{\textsc{Threshold}}_{n-1,k} \right\|\right)$$

whence the overall size is polynomial. It remains to show that $su(D_n) = \textsc{Majority}_n$.

"$\supseteq$": *Let $M \in \textsc{Majority}_n$. We have to show $M \in su(D_n)$, that is, $a \in M$ iff $M \models \varphi_a$ for all $a \in A_n$. For $a \in M$, it is immediate that $M \models \varphi_a$, so let $a_j \notin M$ for some $j \in \{1, \ldots, n\}$. We have to show $M \not\models \varphi_{a_j}$. Since $M \in \textsc{Majority}_n$, we have $|M| = m$ for $k = \lceil \frac{n}{2} \rceil \leq m \leq n-1$ and $M \in \textsc{Threshold}_{n-1,k}$, that is, we have*

$$M \models \Phi^{\textsc{Threshold}}_{n-1,k}(a_1, \ldots, a_{j-1}, a_{j+1}, \ldots, a_n)$$

Together with $M \not\models a_j$, it follows that $M \not\models \varphi_{a_j}$.

"$\subseteq$": *Let* $M \notin \textsc{Majority}_n$. *Then* $|M| = m$ *for* $0 \leq m < \lceil \frac{n}{2} \rceil = k$. *In particular, there is some* $a_j \in A_n \setminus M$. *Now* $m < k$ *implies that there is no* $N \in \textsc{Threshold}_{n-1,k}$ *with* $|N| = m = |M|$. *Thus* $M \not\models \Phi^{\textsc{Threshold}}_{n-1,k}(a_1, \ldots, a_{j-1}, a_{j+1}, \ldots, a_n)$ *whence it follows that* $M \models \varphi_{a_j}$. *Together with* $M \not\models a_j$ *we conclude that* $M \notin su(D_n)$. □

Since every BADF is an ADF of the same size, we get:

Corollary 5.18. $ADF^{su} \not\leq_s NLP^{su}$

In combination with the translation from logic programs to ADFs (implying the relation $\text{NLP}^{su} \leq_s \text{ADF}^{su}$), this means that also ADFs are strictly more succinct than logic programs.

Corollary 5.19. $NLP^{su} <_s ADF^{su}$

BADF vs. AF

It is comparably easy to show that BADF models are strictly more expressive than AFs, since sets of supported models of bipolar ADFs do not have the antichain property.

Proposition 5.20. $AF <_e BADF^{su}$

Proof. Consider vocabulary $A = \{a\}$ *and BADF* $D = (A, \{(a, a)\}, \{\varphi_a\})$ *with* $\varphi_a = a$. *It is straightforward to check that its model set is* $su(D) = \{\emptyset, \{a\}\}$. *Since model sets of AFs under stable extension semantics satisfy the antichain property, there is no equivalent AF over* A. □

This yields the following overall relationships:

$$\text{AF} <_e \text{BADF}^{su} <_e \text{ADF}^{su} \cong_e \text{NLP}^{su} \cong_e \text{CLP}^{su} \cong_e \text{PL}$$

For a concise overview of relative succinctness, we present the results and open problems at a glance in Table 5.2 below.[7]

	BADF^{su}	ADF^{su}	NLP^{su}	PL
BADF^{su}	$=$	$\leq_s$	$\not\leq_s$	$\leq_s$
ADF^{su}	?	$=$	$\not\leq_s$	$\cong_s$
NLP^{su}	?	$<_s$	$=$	$<_s$
PL	?	$\cong_s$	$\not\leq_s$	$=$

Table 5.2: Relative succinctness results for (bipolar) ADFs under the model semantics, normal logic programs under the supported semantics, and classical propositional logic. An entry $\circ$ *in row* $\mathcal{F}_1$ *and column* $\mathcal{F}_2$ *means* $\mathcal{F}_1 \circ \mathcal{F}_2$*; question marks signify open problems.*

[7] We remark that the three open problems in Table 5.2 are really only two: It is easy to show that ADFs and propositional logic behave equivalently in relation to bipolar ADFs, since they are equally expressive and equally succinct; that is, it holds that $\text{ADF}^{su} \leq_s \text{BADF}^{su}$ if and only if $\text{PL} \leq_s \text{BADF}^{su}$.

5.2.2 Stable Semantics

As before, we recall the current state of knowledge:

$$\text{AF} \leq_e \text{BADF}^{st} \leq_e \text{ADF}^{st} <_e \text{PL and AF} \leq_e \text{NLP}^{st} <_e \text{PL} \cong_e \text{CLP}^{st}$$

We first show that BADFs are strictly more expressive than AFs.

Proposition 5.21. *AF $<_e$ BADFst*

Proof. Consider the set $X_2 = \{\{a,b\},\{a,c\},\{b,c\}\}$ of desired models. Dunne et al. (2015) proved that X_2 is not realisable with stable AF semantics. However, the model set X_2 is realisable with BADF D_{X_2} under stable semantics:

$$\varphi_a = \neg b \vee \neg c, \qquad \varphi_b = \neg a \vee \neg c, \qquad \varphi_c = \neg a \vee \neg b$$

Let us exemplarily show that $M = \{a,b\}$ is a stable model (the other cases are completely symmetric): The reduct D^M is characterised by the two acceptance formulas $\varphi_a = \neg b \vee \neg\bot$ and $\varphi_b = \neg a \vee \neg\bot$. We then easily find that $\Gamma_{D^M}(\emptyset,\emptyset) = (M,\emptyset) = \Gamma_{D^M}(M,\emptyset)$. □

Intuitively, the argument for AF non-realisability of X_2 is as follows: Since a and b occur in an extension together, there can be no attack between them. The same holds for the pairs a,c and b,c. But then the set $\{a,b,c\}$ is conflict-free and thus there must be a stable extension containing all three arguments, which is not allowed by X_2. The reason is AFs' restriction to individual attack, as set attack (also called joint or collective attack) suffices to realise X_2 as seen above.

The construction that we used in the proof above to realise X_2 comes from the work of Eiter, Fink, Pührer, Tompits, and Woltran (2013) in logic programming, and can be generalised to realise any non-empty model set satisfying the antichain property.

Definition 5.7. Let $X \subseteq 2^A$. Define the following BADF $D^{st}_X = (A, L, C)$ where C_a for $a \in A$ is given by

$$\varphi_a = \bigvee_{M \in X, a \in M} \left(\bigwedge_{b \in A \setminus M} \neg b \right)$$

and thus $L = \{(b,a) \mid M \in X, a \in M, b \in A \setminus M\}$. ◇

The next result shows that the construction indeed works.

Theorem 5.22. *Let X with $\emptyset \neq X \subseteq 2^A$ be a $\subseteq$-antichain. We find that $st(D^{st}_X) = X$.*

Proof. Let $M \subseteq A$.

"$\subseteq$": *Let $M \notin X$. We show that $M \notin su(D^{st}_X) \supseteq st(D^{st}_X)$; we use a case distinction.*

1. *There is an $N \in X$ with $M \subsetneq N$. Then there is an $a \in N \setminus M$. Consider its acceptance formula φ_a. Since $a \in N$ and $N \in X$, the formula φ_a has a disjunct $\psi_{a,N} = \bigwedge_{b \in A \setminus N} \neg b$. Now $M \subseteq N$ implies $A \setminus N \subseteq A \setminus M$ and M is a model for $\psi_{a,N}$. Thus M is a model for φ_a although $a \notin M$, hence $M \notin su(D^{st}_X)$.*
2. *For all $N \in X$, we have $M \not\subseteq N$. Then $X \neq \emptyset$ implies $M \neq \emptyset$, so let $a \in M$. For each $N \in X$ with $a \in N$, the acceptance formula φ_a contains a disjunct $\psi_{a,N} = \bigwedge_{b \in A \setminus N} \neg b$. By assumption, for each $N \in X$ there is a $b_N \in M \setminus N$. Clearly $b_N \in A \setminus N$ and b_N is evaluated to true by M. Hence for each $N \in X$ with $a \in N$, the disjunct $\psi_{a,N}$ is evaluated to false by M. Thus φ_a is false under M and $M \notin su(D^{st}_X)$.*

"$\supseteq$": *Let $M \in X$. We first show that $M \in su(D^{st}_X)$, that is: for all $a \in A$, we find $a \in M$ iff M is a model for φ_a.*

1. *Let $a \in M$. By construction, we have that φ_a in D^{st}_X contains a disjunct of the form $\psi_{a,M} = \bigwedge_{b \in A \setminus M} \neg b$. According to the interpretation M, all such $b \in A \setminus M$ are false and thus $\psi_{a,M}$ is true whence φ_a is true.*
2. *Let $a \in A \setminus M$ and consider its acceptance formula φ_a. Assume to the contrary that M is a model for φ_a. Then there is some $N \in X$ with $a \in N$ such that M is a model for $\psi_{a,N} = \bigwedge_{b \in A \setminus N} \neg b$, that is, $A \setminus N \subseteq A \setminus M$. Hence $M \subseteq N$; and, since $a \in N \setminus M$, even $M \subsetneq N$, whence X is not a $\subseteq$-antichain. Contradiction. Thus M is no model for φ_a.*

Now consider the reduct D^M of D^{st}_X with respect to M. There, φ^M_a contains the disjunct $\psi^M_{a,M} = \psi_{a,M}[b/\bot : b \notin M]$ where all $b \in A \setminus M$ have been replaced by false, whence $\psi^M_{a,M} = \neg\bot \wedge \ldots \wedge \neg\bot$ and φ^M_a is equivalent to true. Thus each $a \in M$ is true in the least fixpoint of Γ_{D^M} and thus $M \in st(D^{st}_X)$. □

The restriction to non-empty model sets is immaterial for relative expressiveness, since we can use the construction of Theorem 5.2 and the fact that $st(D) \subseteq su(D)$ for any ADF D to realise the empty model set. As the stable model semantics for ADFs and logic programs both have the antichain property, we get:

Corollary 5.23. *$ADF^{st} \leq_e BADF^{st}$ and $NLP^{st} \leq_e BADF^{st}$*

This leads to the following overall relationships:

$$\text{AF} <_e \text{BADF}^{st} \cong_e \text{ADF}^{st} \cong_e \text{NLP}^{st} <_e \text{PL} \cong_e \text{CLP}^{st}$$

We remark that the antichain property provides a *characterisation* of realisability with the stable semantics; that is, a model set is stable-realisable iff it is a $\subseteq$-antichain.

5.2.3 Supported vs. Stable Semantics

Now we put the supported and stable pictures together. From the proof of Theorem 5.22, we can read off that for the canonical realisation D^{st}_X of an antichain X, the supported and stable semantics coincide, that is, $su(D^{st}_X) = st(D^{st}_X) = X$. With this observation, also bipolar ADFs under the supported semantics can realise any antichain, and we have this:

Proposition 5.24. *$BADF^{st} \leq_e BADF^{su}$*

As we have seen in Proposition 5.20, there are bipolar ADFs with supported-model sets that are not antichains. We get:

Corollary 5.25. *$BADF^{st} <_e BADF^{su}$*

This result allows us to close the last gap and put together the big picture on relative expressiveness in Figure 5.2 below.

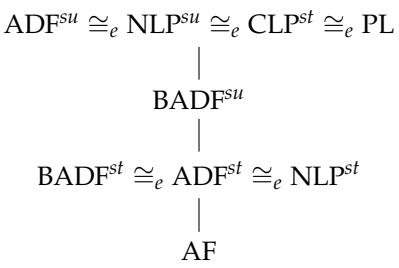


Figure 5.2: The expressiveness hierarchy. Expressiveness strictly increases from bottom to top. $\mathcal{F}^\sigma$ denotes formalism $\mathcal{F}$ under semantics σ, where "su" is the supported and "st" the stable model semantics; formalisms are among AFs (argumentation frameworks), ADFs (abstract dialectical frameworks), BADFs (bipolar ADFs), NLPs (normal logic programs), CLPs (canonical logic programs – normal logic programs extended by double negation as failure) and PL (propositional logic).

5.3 Allowing Vocabulary Expansion

Up to here, we only considered *compact* realisations, that do not introduce new vocabulary elements. In this section, we allow the introduction of a small number of new atoms/arguments/statements. More precisely, *small* means the number is linear in the size of the source knowledge base (representing the model set that we wish to realise in a target language). For the purpose of realisability, the new vocabulary elements are projected out of the resulting models.

As it turns out, adding additional arguments already makes AFs universally expressive (under projection). More technically, we will now show that for each propositional formula φ over vocabulary A, there exists an AF F_φ over an expanded vocabulary $A \cup A_\varphi$ such that the models of φ and the stable extensions of F_φ correspond one-to-one. Roughly, this is possible since AFs can be regarded as a syntactic variant of classical propositional logic that has as its only connective the logical NOR "$\downarrow$" (Gabbay, 2011; Brewka et al., 2011). Using this connective, negation is expressed by $\neg\varphi = \varphi \downarrow \varphi$ and disjunction by $\varphi \vee \psi = \neg(\varphi \downarrow \psi) = (\varphi \downarrow \psi) \downarrow (\varphi \downarrow \psi)$. These equivalences can be used to translate arbitrary propositional formulas (over $\neg, \wedge, \vee$) into the syntactical $\downarrow$-fragment; to guarantee that the size increase is at most linear, we introduce names a_ψ for subformulas ψ (Tseitin, 1968). The next definition combines all of these ideas.

Definition 5.8. Let φ be a formula using $\neg, \wedge, \vee$ over vocabulary A. Define the sets A_φ and R_φ inductively as follows:

$$
\begin{aligned}
A_\top &= \{a_\top\} & R_\top &= \emptyset \\
A_\bot &= \{a_\bot\} & R_\bot &= \{(a_\bot, a_\bot)\} \\
A_p &= \{p, a_{\neg p}\} \text{ for } p \in A & R_p &= \{(p, a_{\neg p}), (a_{\neg p}, p)\} \text{ for } p \in A \\
A_{\neg\xi} &= \{a_{\neg\xi}\} \cup A_\xi & R_{\neg\xi} &= \{(a_\xi, a_{\neg\xi})\} \cup R_\xi \\
A_{\zeta\wedge\xi} &= \{a_{\zeta\wedge\xi}, a_{\neg\zeta}, a_{\neg\xi}\} \cup A_{\neg\zeta} \cup A_{\neg\xi} & R_{\zeta\wedge\xi} &= \{(a_{\neg\zeta}, a_{\zeta\wedge\xi}), (a_{\neg\xi}, a_{\zeta\wedge\xi})\} \cup R_{\neg\zeta} \cup R_{\neg\xi} \\
A_{\zeta\vee\xi} &= \{a_{\zeta\vee\xi}, a_{\zeta\downarrow\xi}\} \cup A_\zeta \cup A_\xi & R_{\zeta\vee\xi} &= \{(a_{\zeta\downarrow\xi}, a_{\zeta\vee\xi}), (a_\zeta, a_{\zeta\downarrow\xi}), (a_\zeta, a_{\zeta\downarrow\xi})\} \cup R_\zeta \cup R_\xi
\end{aligned}
$$

The *AF associated to* φ is given by $F_\varphi = (A_\varphi \cup A_\bot, R_\varphi \cup \{(a_\varphi, a_\bot)\} \cup R_\bot)$. ◇

The argument $a_\top$ is unattacked and thus part of every stable extension (is true in every interpretation); the argument $a_\bot$ attacks itself and thus cannot be part of any stable extension (is

false in every interpretation). The mutually attacking arguments p and $a_{\neg p}$ for $p \in A$ serve to "guess" a valuation of A, while a_φ and $a_\bot$ guarantee that only (and all) valuations that are models of φ can lead to stable extensions of F_φ: intuitively, $a_\bot$ must be attacked, and the only candidate to do so is a_φ. The arguments and attacks for the Boolean connectives express their usual truth-theoretic semantics, as our first technical result for this translation shows.

Lemma 5.26. *Let φ be a formula over vocabulary A and F_φ its associated AF. For each stable extension M of F_φ and $a_\zeta, a_\xi \in A_\varphi$, we have:*

- *$a_{\neg\xi} \in M$ iff $a_\xi \notin M$;*
- *$a_{\zeta\wedge\xi} \in M$ iff both $a_\zeta \in M$ and $a_\xi \in M$;*
- *$a_{\zeta\vee\xi} \in M$ iff one of $a_\zeta \in M$ or $a_\xi \in M$;*
- *$a_{\zeta\downarrow\xi} \in M$ iff neither $a_\zeta \in M$ nor $a_\xi \in M$.*

Proof.
- *By definition, the only attacker of an argument of the form $a_{\neg\xi}$ is the argument a_ξ. Thus $a_\xi \in M$ iff $a_{\neg\xi} \notin M$.*
- *The only attackers of $a_{\zeta\wedge\xi}$ are the arguments $a_{\neg\zeta}$ and $a_{\neg\xi}$. By the case above, we have $a_{\neg\zeta} \in M$ iff $a_\zeta \notin M$, and $a_{\neg\xi} \in M$ iff $a_\xi \notin M$. Consequently, $a_\zeta \in M$ and $a_\xi \in M$ iff $a_{\neg\zeta} \notin M$ and $a_{\neg\xi} \notin M$ iff $a_{\zeta\wedge\xi} \in M$.*
- *The only attacker of $a_{\zeta\vee\xi}$ is the argument $a_{\zeta\downarrow\xi}$. Similarly to the previous cases, we can show that $a_{\zeta\downarrow\xi} \in M$ iff $a_\zeta \notin M$ and $a_\xi \notin M$, and that $a_{\zeta\vee\xi} \in M$ iff $a_{\zeta\downarrow\xi} \notin M$. In combination, $a_{\zeta\vee\xi} \in M$ iff $a_\zeta \in M$ or $a_\xi \in M$.*
- *The only attackers of $a_{\zeta\downarrow\xi}$ are the arguments a_ζ and a_ξ. It directly follows that $a_{\zeta\downarrow\xi} \in M$ iff neither $a_\zeta \in M$ nor $a_\xi \in M$.* □

These correspondences can be used to show by induction that the newly introduced arguments capture the semantics of the formulas they encode (for all subformulas ψ of φ).

Lemma 5.27. *Let φ be a formula over A and F_φ its associated AF. For each stable extension M of F_φ and $a_\psi \in A_\varphi$, we have $a_\psi \in M$ iff $M \cap A$ is a model of ψ.*

Proof. Let M be a stable extension of F. We use structural induction on ψ.

$\psi = \top$: *Trivial: $a_\top \in M$ since it has no attackers.*

$\psi = \bot$: *Trivial: $a_\bot \notin M$ since the set $\{a_\bot\}$ is not conflict-free.*

$\psi = p \in A$: *Trivial: $p \in M$ iff $M \models p$ by definition.*

$\psi = \neg\xi$: *$a_\psi \in M$ iff $a_{\neg\xi} \in M$ iff $a_\xi \notin M$ iff $M \not\models \xi$ iff $M \models \neg\xi$ iff $M \models \psi$.*

$\psi = \zeta \wedge \xi$: *$a_\psi \in M$ iff $a_{\zeta\wedge\xi} \in M$ iff $a_\zeta \in M$ and $a_\xi \in M$ iff $M \models \zeta$ and $M \models \xi$ iff $M \models \zeta \wedge \xi$ iff $M \models \psi$.*

$\psi = \zeta \vee \xi$: *$a_\psi \in M$ iff $a_{\zeta\vee\xi} \in M$ iff $a_\zeta \in M$ or $a_\xi \in M$ iff $M \models \zeta$ or $M \models \xi$ iff $M \models \zeta \vee \xi$ iff $M \models \psi$.*

$\psi = \zeta \downarrow \xi$: *$a_\psi \in M$ iff $a_{\zeta\downarrow\xi} \in M$ iff $a_\zeta \notin M$ and $a_\xi \notin M$ iff $M \not\models \zeta$ and $M \not\models \xi$ iff $M \models \zeta \downarrow \xi$ iff $M \models \psi$.* □

This lets us show the main result of this section, namely that the AF stable extension semantics is universally expressive under projection.

Theorem 5.28. *Let φ be a formula over vocabulary A and F_φ its associated AF.*

1. *For each model $M \subseteq A$ of φ, there exists a stable extension E of F_φ with $M = E \cap A$.*
2. *For each stable extension E of F_φ, the set $E \cap A$ is a model of φ.*

Proof. 1. *Let $M \subseteq A$ be a model of φ. Define the set*

$$E = \{a_\psi \mid a_\psi \in A_\varphi, M \models \psi\}$$

Observe that $M = E \cap A$. By presumption, $a_\varphi \in E$. It remains to show that E is a stable extension, that is, E is conflict-free and attacks all arguments $b \notin E$.

E is conflict-free: *Assume to the contrary that there is an attack $r = (a, b) \in R_\varphi$ with $a, b \in E$. By definition, there are only these cases:*

- *a is arbitrary and $b = \bot$. But then by definition of E we get $M \models \bot$, contradiction.*
- *$r = (p, a_{\neg p})$ or $r = (a_{\neg p}, p)$ for $p \in A$. But then by definition of E we get $M \models p$ and $M \models \neg p$, contradiction.*
- *$r = (a_\xi, a_{\neg\xi})$. But then by definition of E we get $M \models \xi$ and $M \models \neg\xi$, contradiction.*
- *$r = (a_{\neg\zeta}, a_{\zeta\wedge\xi})$ or $r = (a_{\neg\xi}, a_{\zeta\wedge\xi})$. Then $M \models \zeta \wedge \xi$, and $M \models \neg\zeta$ or $M \models \neg\xi$, contradiction.*
- *$r = (a_{\zeta\downarrow\xi}, a_{\zeta\vee\xi})$. Then $M \models \zeta \downarrow \xi$, whence $M \models \neg(\zeta \vee \xi)$. But also $M \models \zeta \vee \xi$, contradiction.*
- *$r = (a_\zeta, a_{\zeta\downarrow\xi})$ or $r = (a_\xi, a_{\zeta\downarrow\xi})$. Then $M \models \zeta \downarrow \xi$, and $M \models \zeta$ or $M \models \xi$. But then also $M \models \zeta \vee \xi$, contradiction.*

E attacks all arguments not in E: *Let $b \in (A \cup A_\varphi \cup \{a_\bot\}) \setminus E$ be an argument. By definition, there is a formula ψ such that $b = a_\psi$ and $M \not\models \psi$. We use structural induction.*

- *If $\psi = \bot$ then $a_\varphi \in E$ attacks $a_\bot$ by definition.*
- *If $\psi = \neg\xi$, then $M \models \xi$ whence $a_\xi \in E$ attacks a_ψ by definition.*
- *If $\psi = \zeta \wedge \xi$, then $M \models \neg\zeta$ or $M \models \neg\xi$ whence $a_{\neg\zeta} \in E$ or $a_{\neg\xi} \in E$. In any case, E attacks a_ψ by definition.*
- *If $\psi = \zeta \vee \xi$, then $M \models \zeta \downarrow \xi$ whence $a_{\zeta\downarrow\xi} \in E$ attacks a_ψ by definition.*
- *If $\psi = \zeta \downarrow \xi$, then $M \models \zeta \vee \xi$ whence $a_\zeta \in E$ or $a_\xi \in E$.*

In any case, E attacks a_ψ by definition.

2. *Let E be a stable extension of F_φ. Since E is conflict-free, $a_\bot \notin E$. Since E is stable, E attacks $a_\bot$, which yields $a_\varphi \in E$. By Lemma 5.27, $E \cap A$ is a model of φ.* □

In particular, F_φ has no stable extension iff φ is unsatisfiable. While this shows that the construction of Definition 5.8 works as intended, it remains to show that the number of new arguments is at most linear in the formula size. We can even show that the total increase in size is only linear, thus also the number of new arguments is linear.

Proposition 5.29. *For any formula φ, we find that $\|F_\varphi\| \in O(\|\varphi\|)$.*

Proof. We first note that

$$\begin{aligned}\|F_\varphi\| &= \|(A_\varphi \cup A_\bot, R_\varphi \cup \{(a_\varphi, a_\bot)\} \cup R_\bot)\| \\ &= |A_\varphi \cup A_\bot| + |R_\varphi \cup \{(a_\varphi, a_\bot)\} \cup R_\bot| \\ &= |A_\varphi| + 1 + |R_\varphi| + 2 \\ &= |A_\varphi| + |R_\varphi| + 3\end{aligned}$$

We now use structural induction on φ to show that for all formulas φ, we find $|A_\varphi| \leq 5 \cdot \|\varphi\|$ and $|R_\varphi| \leq 4 \cdot \|\varphi\|$. It then follows that $\|F_\varphi\| \leq (5+4) \cdot \|\varphi\| + 3 = 9 \cdot \|\varphi\| + 3 \in O(\|\varphi\|)$.

$\varphi = \top$:

$$\begin{aligned}|A_\top| &= |\{a_\top\}| = 1 \leq 5 = 5 \cdot \|\top\| \\ |R_\top| &= |\emptyset| = 0 \leq 4 = 4 \cdot \|\top\|\end{aligned}$$

$\varphi = \bot$:

$$\begin{aligned}|A_\bot| &= |\{a_\bot\}| = 1 \leq 5 = 5 \cdot \|\bot\| \\ |R_\bot| &= |\{(a_\bot, a_\bot)\}| = 1 \leq 4 = 4 \cdot \|\bot\|\end{aligned}$$

$\varphi = a \in A$:

$$\begin{aligned}|A_a| &= |\{a, a_{\neg a}\}| = 2 \leq 5 = 5 \cdot \|a\| \\ |R_a| &= |\{(a, a_{\neg a}), (a_{\neg a}, a)\}| = 2 \leq 4 = 4 \cdot \|a\|\end{aligned}$$

$\varphi = \neg\xi$:

$$\begin{aligned}|A_\varphi| &= |A_\xi \cup \{a_{\neg\xi}\}| \leq |A_\xi| + 1 \leq (5 \cdot \|\xi\|) + 1 \leq 5 \cdot (\|\xi\| + 1) = 5 \cdot \|\varphi\| \\ |R_\varphi| &= |R_\xi \cup \{(a_\xi, a_{\neg\xi})\}| \leq |R_\xi| + 1 \leq (4 \cdot \|\xi\|) + 1 \leq 4 \cdot (\|\xi\| + 1) = 4 \cdot \|\varphi\|\end{aligned}$$

$\varphi = \zeta \wedge \xi$:

$$\begin{aligned}|A_\varphi| &\leq |A_{\neg\zeta}| + |A_{\neg\xi}| + 3 \leq (|A_\zeta| + 1) + (|A_\xi| + 1) + 3 \\ &\leq (5 \cdot \|\zeta\| + 1) + (5 \cdot \|\xi\| + 1) + 3 = 5 \cdot \|\zeta\| + 5 \cdot \|\xi\| + 5 \\ &= 5 \cdot (\|\zeta\| + \|\xi\| + 1) = 5 \cdot \|\zeta \wedge \xi\|\end{aligned}$$

$$\begin{aligned}|R_\varphi| &\leq |R_{\neg\zeta}| + |R_{\neg\xi}| + 2 \leq (|R_\zeta| + 1) + (|R_\xi| + 1) + 2 \\ &\leq (4 \cdot \|\zeta\| + 1) + (4 \cdot \|\xi\| + 1) + 2 = 4 \cdot \|\zeta\| + 4 \cdot \|\xi\| + 4 \\ &= 4 \cdot (\|\zeta\| + \|\xi\| + 1) = 4 \cdot \|\varphi\|\end{aligned}$$

$\varphi = \zeta \vee \xi$:

$$\begin{aligned}|A_\varphi| &\leq |A_\zeta| + |A_\xi| + 2 \leq 5 \cdot \|\zeta\| + 5 \cdot \|\xi\| + 2 \\ &\leq 5 \cdot \|\zeta\| + 5 \cdot \|\xi\| + 5 = 5 \cdot (\|\zeta\| + \|\xi\| + 1) = 5 \cdot \|\varphi\|\end{aligned}$$

$$\begin{aligned}|R_\varphi| &\leq |R_\zeta| + |R_\xi| + 3 \leq (4 \cdot \|\zeta\|) + (4 \cdot \|\xi\|) + 3 \\ &\leq 4 \cdot \|\zeta\| + 4 \cdot \|\xi\| + 4 = 4 \cdot (\|\zeta\| + \|\xi\| + 1) = 4 \cdot \|\varphi\|\end{aligned}$$

□

Hence under projection, the AF stable extension semantics can realise as much as propositional logic can. With the results of the previous section (AF $\leq_e$ PL), this means that allowing to introduce a linear number of new vocabulary elements (that are later projected out), all languages considered in this chapter are equally (universally) expressive.

However, we must note that equal expressiveness does not mean equal efficiency: When we assume that a knowledge base of size n leads to a search space of size $O(2^n)$, then a *linear* increase in knowledge base size (that is, from n to $c \cdot n$ for some constant c) leads to a *polynomial* increase in search space size (that is, from $O(2^n)$ to $O(2^{c \cdot n}) = O\big((2^n)^c\big)$.

5.4 Discussion

We compared the expressiveness of abstract argumentation frameworks, abstract dialectical frameworks, normal (and canonical) logic programs and propositional logic. We showed that expressiveness under different semantics varies for the formalisms and obtained a neat expressiveness hierarchy. These results inform us about the capabilities of these languages to encode sets of two-valued interpretations, and help us decide which languages to use for specific applications. Furthermore, we have seen that the results are sensitive to the vocabulary one is permitted to use, as the hierarchy collapses when we allow to introduce even only a linear number of new atoms.

Concerning succinctness, we have shown that ADFs (under model semantics) are exponentially more succinct than normal logic programs (under supported model semantics), and that even bipolar ADFs (under model semantics) – although being less expressive – can succinctly express some model sets where equivalent normal logic programs (under supported model semantics) over the same vocabulary must necessarily blow up exponentially in size. It is open whether the converse direction also holds, that is, whether BADFs are exponentially more succinct than logic programs (if $\text{NLP}^{su} \leq_s \text{BADF}^{su}$) or the two are just mutually incomparable in terms of succinctness (if $\text{NLP}^{su} \not\leq_s \text{BADF}^{su}$). For the stable semantics, relative succinctness of logic programs and BADFs is completely open, partly due to the technical aspect that the two stable semantics are conceptually different, as ADFs in fact employ *ultimate* stable models (Section 3.3). Furthermore, for general ADFs, the computational complexity of the model existence problem of stable semantics is higher than for normal logic programs, so a succinctness comparison with regard to stable models would be of limited significance.

It is easy to see that AFs have a somewhat special role as they are representationally succinct in any case: for a vocabulary A_n, there is syntactically no possibility to specify a knowledge base (an AF) of exponential size, since the largest AF over A_n has size $\|(A_n, A_n \times A_n)\| = n + n^2$ and is thus polynomially large. So anything that can be expressed with an AF can be expressed in reasonable space by definition. However, this "strength" of AFs should be taken with a grain of salt, since they are comparably inexpressive. This can (in addition to the results we presented) already be seen from a simple counting argument: even if all syntactically different AFs over A_n were semantically different (which they are not), they could express at most 2^{n^2} different model sets, which is – for increasing n – negligible in relation to the 2^{2^n} possible model sets over A_n.

In their original paper, Gogic et al. (1995) also used a relaxed version of succinctness, where they allowed to introduce a linear number of new variables. It follows from our results in Section 5.3 that all formalisms we consider here are equally succinct under this relaxed notion.

Recently, Shen and Zhao (2014) showed that canonical logic programs under stable models and propositional logic are succinctly incomparable (under the assumption that $\mathsf{P} \not\subseteq \mathsf{NC}^1_{/\mathsf{poly}}$, the Boolean circuit equivalent of the assumption $\mathsf{NP} \not\subseteq \mathsf{P}$), and also provide interesting avenues

for further succinctness studies. We can also add succinctness questions of our own: firstly that of comparing *disjunctive* logic programs under stable models with general ADFs under stable models, since the two have an equally complex (Σ_2^P-complete) model existence problem (Eiter and Gottlob, 1995; Theorem 4.39). What is more, as the reader knows there are alternative proposals for stable model semantics for ADFs:

- the approximate stable model semantics (Definition 3.2; Strass, 2013, Definition 3.2), for which model existence is NP-complete and thus potentially easier than that of ultimate stable model semantics;
- the "grounded model" semantics by Bogaerts, Vennekens, and Denecker (2015, Definition 6.8), whose model existence problem is also Σ_2^P-complete (Bogaerts et al., 2015);
- the "F-stable model" semantics by Alviano and Faber (2015, Definition 10).

It follows from Theorem 5.9 of Bogaerts et al. (2015) that grounded models and F-stable models coincide. Still, they are demonstrably different from both approximate and ultimate stable models for ADFs (Alviano and Faber, 2015),[8] and their relative succinctness in comparison to normal/disjunctive logic programs is unanalysed.

There is more potential for further work on expressiveness and succinctness. First of all, a "nice" characterisation of bipolar ADF realisability is still missing; we are unsure whether much improvement over Proposition 5.11 is possible. Incidentally, for AFs the exact characterisation of compact stable extension realisability constitutes a major open problem (Baumann et al., 2014; Dunne et al., 2015; Baumann et al., 2016). Second, there are further semantics for abstract dialectical frameworks whose expressiveness could be studied; Dunne et al. (2015) and Dyrkolbotn (2014) already analyse many of them for argumentation frameworks. This chapter is thus only a start and the same can be done for the remaining semantics. For example the admissible, complete and preferred semantics are all defined for AFs, (B)ADFs and NLPs (see Chapter 3), and Pührer (2015) has already made a huge step into that direction by characterising realisability. Third, there are further formalisms in abstract argumentation (Brewka et al., 2014) whose expressiveness is by and large unexplored to the best of our knowledge. Finally, the representational succinctness of the subclass of bipolar ADFs (using bipolar propositional formulas to represent them) under supported model semantics is mostly open (cf. Table 5.2), with some evidence pointing toward meaningful capabilities.

[8] In the terminology of Alviano and Faber (2015), approximate stable models are called S-stable models and ultimate stable models (Brewka et al., 2013) are called B-stable models. Both are shown to be different from F-stable models.

Chapter 6

An Application to Theory Bases

Abstract argumentation frameworks (AFs; Dung, 1995) are widely used in argumentation research. As we have seen in the background chapter (Chapter 2), such an AF consists of a set of arguments and an attack relation between these arguments. Their semantics determines which sets of arguments of a given AF can be accepted according to specific criteria. A common way to employ Dung's AFs is as abstraction formalism. In this view, expressive languages are used to model concrete argumentation scenarios, and translations into Dung AFs provide these original languages with semantics. The advantage of translating into an argumentation formalism is that the resulting semantics can be given a dialectical interpretation, which can be used to inform humans how a particular conclusion was inferred.

However, the approach is not without its problems. Caminada and Amgoud (2007) reported some difficulties they encountered when defining an abstract argumentation-based semantics for defeasible theory bases. Defeasible theory bases are simple logic-inspired formalisms working with inference rules on a set of literals. Inference rules can be strict, in which case the conclusion of the inference (a literal) must necessarily hold whenever all antecedents (also literals) hold. Inference rules can also be defeasible, which means that the conclusion *usually* holds whenever the antecedents hold. Here, the word "usually" suggests that there could be exceptional cases where a defeasible rule has not been applied (Pollock, 1987).

In response to the problems they encountered, Caminada and Amgoud (2007) stated general rationality postulates for AFs based on defeasible theories. The intention of these postulates is to mathematically capture what humans perceive as rational behaviour from the semantics of defeasible theory bases. First of all the *closure* postulate says that whatever model or extension the target formalism (the AF) produces, it must be closed under application of strict rules, meaning that all applicable strict rules have been applied. Direct and indirect *consistency* postulates express that any model or extension of the target formalism must be internally consistent with respect to the literals of the defeasible theory base (directly) and even with respect to application of strict rules (indirectly).

Later, Wyner et al. (2009) criticised Caminada and Amgoud's definition of arguments on ontological grounds and gave an alternative translation. We are agnostic with respect to Wyner et al.'s criticism, but use their translation as a starting point for our own work. Such a further refinement is necessary since the translation of Wyner et al. (2009) still yields unintuitive results on benchmark examples and does not satisfy the closure and indirect consistency postulates. Wyner et al. (2013) later fixed these issues by adding a meta-level integrity constraint on the obtained extensions, thus ruling out violation of the postulates. Our translation has this integrity constraint built into it, such that models can be taken as they are. In more recent

further work, Wyner, Bench-Capon, Dunne, and Cerutti (2015) once again modified the definition of AFs associated to defeasible theories – unfortunately in such a way that the rationality postulates are again not fulfilled.[1] Therefore, we will mainly base this chapter on the work of Wyner et al. (2013).

The basis of our solution to the aforementioned problems of previous approaches is a shift in the target language. While until now abstract argumentation frameworks were the formalism of choice, we will use the more general abstract *dialectical* frameworks (ADFs) (Brewka and Woltran, 2010). Where AFs allow only attacks between arguments, ADFs can also represent support relations and many more. The modelling capacities of ADFs in comparison to AFs – which we studied earlier in this thesis – enables us to give a direct and straightforward translation from defeasible theory bases to abstract dialectical frameworks. We will show that this translation – the first main contribution of this chapter – treats the benchmark examples right and satisfies the rationality postulates of Caminada and Amgoud (2007). We consider this further important evidence that abstract dialectical frameworks are useful tools for representing and reasoning about argumentation scenarios. We also perform a complexity analysis of our translation; this is significant in that we are not aware of complexity analyses of the mentioned previous approaches.

The availability of support in ADFs (in contrast to AFs) as a target formalism will be of fundamental importance to our translation. Among other things, it will allow us to resolve cyclic dependencies among literals in a defeasible theory base in a straightforward way. The treatment of such support cycles is built into ADF standard semantics, which can be considered a product of decades of research into nonmonotonic knowledge representation languages.

As the second main contribution of this chapter, we introduce a possible-worlds semantics for defeasible theory bases. This provides a language for formulating different intuitions about the meaning of strict and defeasible rules. Furthermore, it nicely illustrates the difficulties in formally defining semantics for collections of such rules. The semantics is inspired by possible-worlds semantics for autoepistemic logic (Denecker et al., 2003), we therefore indirectly present potential epistemic modal readings of strict and defeasible rules.

In the rest of this chapter, we first recall some necessary background on defeasible theory bases. In Section 6.2 we look at the translations of Caminada and Amgoud (2007) and Wyner et al. (2009), discuss some problems of these, and introduce generalised versions of the rationality postulates. In Section 6.3 we then define our own translation from defeasible theory bases to ADFs. We show how it treats the problematic examples, prove that it satisfies the (generalised versions of the) rationality postulates and analyse its computational complexity. We then introduce our direct semantics for defeasible theories and illustrate its behaviour on several examples, and afterwards clarify its connections to autoepistemic logic. We conclude with a discussion of related and future work.

6.1 Background on Defeasible Theories

Following Caminada and Amgoud (2007), we use a set *Lit* of literals that are built using syntactical negation $\neg\cdot$ and define a semantic negation function $\bar{\cdot}$ such that for an atom p we have $\overline{p} = \neg p$ and $\overline{\neg p} = p$. Throughout this chapter, we assume that *Lit* is closed under negation in the sense that $\psi \in Lit$ implies $\overline{\psi} \in Lit$. A set $S \subseteq Lit$ of literals is *consistent* iff there is no literal $\psi \in Lit$ such that both $\psi \in S$ and $\neg\psi \in S$. For literals $\phi_1, \ldots, \phi_n, \psi \in Lit$, a *strict rule* over *Lit* is of the form $r : \phi_1, \ldots, \phi_n \rightarrow \psi$; a *defeasible rule* over *Lit* is of the form $r : \phi_1, \ldots, \phi_n \Rightarrow \psi$. (The

[1] The strict rule set $\{\rightarrow a, \quad a \rightarrow \neg a\}$ is "well-formed" according to their definition, consequently their approach violates closure as not both a and $\neg a$ are concluded from the resulting AF.

only difference is the arrows.) Here r is the unique *rule name*, the literals $\phi_1, \ldots, \phi_n$ constitute the *rule body* and ψ is the *rule head* or *conclusion*. Intuitively, a strict rule says that the rule head is necessarily true whenever all body literals are true; a defeasible rule says that the head ψ is *usually* true whenever all body literals are true. In definitions, we use the symbol $\Rightarrow$ as meta-level variable for $\rightarrow$ and $\Rightarrow$.

For a set $M \subseteq Lit$ of literals and a set *StrInf* of strict rules over *Lit*, we say that M is *closed under StrInf* iff $r : \phi_1, \ldots, \phi_n \rightarrow \psi \in StrInf$ and $\phi_1, \ldots, \phi_n \in M$ imply $\psi \in M$. Accordingly, the *closure of M under StrInf* is the $\subseteq$-least set $Cl_{StrInf}(M)$ that contains M and is closed under *StrInf*. A *defeasible theory base* (also *defeasible theory* or *theory base*) is a triple $(Lit, StrInf, DefInf)$ where *Lit* is a set of literals, *StrInf* is a set of strict rules over *Lit* and *DefInf* is a set of defeasible rules over *Lit*. The semantics of theory bases is usually defined via a translation to abstract argumentation frameworks. Here, we will in a similar fashion define such a translation to ADFs. It will turn out that ADFs resulting from our automatic translation from defeasible theory bases are all bipolar. This is especially significant as the complexity results of Chapter 4 show that bipolar ADFs are as complex as AFs, thus the additional modelling capacities of bipolar ADFs come essentially for free.

6.2 Instantiations to Abstract Argumentation Frameworks

The general approach to provide a semantics for defeasible theories is to translate the defeasible theory into an argumentation formalism and then let the already existing semantics for that argumentation formalism determine the semantics of the defeasible theory. In the literature, the target formalism of choice are Dung's abstract argumentation frameworks. They abstract away from everything except arguments and attacks between them, so to define a translation to AFs one has to define arguments and attacks. We now review two particular such approaches.

6.2.1 The Approach of Caminada and Amgoud (2007)

Caminada and Amgoud (2007) define a translation from defeasible theories to argumentation frameworks. They create arguments in an inductive way by applying one or more inference rules. The internal structure of the arguments reflects how a particular conclusion was derived by applying an inference rule to the conclusions of subarguments, and allows arguments to be nested. So the base case of the induction takes into account rules with empty body, that is, rules of the form $\rightarrow \psi$ (or $\Rightarrow \psi$) for some literal ψ. Each such rule leads to an argument $A = [\rightarrow \psi]$ (or $[\Rightarrow \psi]$), and the conclusion of the rule becomes the conclusion of the argument. For the induction step, we assume there are arguments $A_1, \ldots, A_n$ with conclusions $\phi_1, \ldots, \phi_n$, respectively. If there is a strict rule $\phi_1, \ldots, \phi_n \rightarrow \psi$, we can build a new argument $A = [A_1, \ldots, A_n \rightarrow \psi]$ with conclusion ψ. (Likewise, if there is a defeasible rule $\phi_1, \ldots, \phi_n \Rightarrow \psi$, we can build a new argument $A = [A_1, \ldots, A_n \Rightarrow \psi]$.) Similar to rules, arguments can be strict or defeasible, where application of at least one defeasible rule makes the whole argument defeasible. In other words, strict arguments only use strict rules to derive their conclusion.

For these arguments, Caminada and Amgoud (2007) then define two different kinds of attacks, rebuts and undercuts. An argument a rebuts another argument b if a subargument of a concludes some literal ψ, while there is a defeasible subargument of b that concludes $\overline{\psi}$. An argument a undercuts another argument b if the latter has a subargument that results from applying a defeasible rule and the applicability of that rule is disputed by a subargument of

a. (So as a matter of principle, only defeasible arguments can be attacked.) Caminada and Amgoud (2007) observed some difficulties of this translation.

Example 6.1 (Married John. Caminada and Amgoud, 2007, Example 4). Consider the following vocabulary with intended natural-language meaning:

w ... John wears something that looks like a wedding ring,
g ... John often goes out late with his friends,
m ... John is married,
b ... John is a bachelor,
h ... John has a spouse.

There are several relationships between these propositions, which are captured in the following theory base:

$$\begin{aligned} \mathit{Lit} &= \{w, g, h, m, b, \neg w, \neg g, \neg h, \neg m, \neg b\} \\ \mathit{StrInf} &= \{r_1 : \rightarrow w, \quad r_2 : \rightarrow g, \quad r_3 : b \rightarrow \neg h, \quad r_4 : m \rightarrow h\} \\ \mathit{DefInf} &= \{r_5 : w \Rightarrow m, \quad r_6 : g \Rightarrow b\} \end{aligned}$$

In the ASPIC system of Caminada and Amgoud (2007), all the literals in the set $S = \{w, g, m, b\}$ are contained in all extensions (with respect to any of Dung's standard semantics) of the constructed AF. Caminada and Amgoud observe that this is clearly unintended since the natural-language interpretation would be that John is a married bachelor. Moreover, the closure of S under StrInf is $\mathit{Cl}_{\mathit{StrInf}}(S) = \{w, g, m, b, h, \neg h\}$, which is inconsistent. So not only are there applicable strict rules that have not been applied in S, but their application would lead to inconsistency. ◇

To avoid anomalies such as the one just seen, Caminada and Amgoud (2007) went on to define three natural rationality postulates for rule-based argumentation-based systems that are concerned with the interplay of consistency and strict rule application. Our formulation of them is slightly different for various reasons:

- We are concerned with argumentation frameworks as well as with abstract dialectical frameworks in this chapter, so we made the postulates parametric in the target argumentation formalism.
- We removed the respective second condition on the sceptical conclusions with respect to all extensions/models. Propositions 4 and 5 in (Caminada and Amgoud, 2007) show that they are redundant in their case.
- We are not constrained to formalisms and semantics where there are only finitely many extensions/models.
- For the sake of readability, we assume that the literals *Lit* of the defeasible theory are contained in the vocabulary of the target formalism.[2]

The first postulate requires that the set of conclusions for any extension should be closed under application of strict rules.

[2]This is not a proper restriction since reconstruction of conclusions about the original defeasible theory is one of the goals of the whole enterprise and so there should be at least a translation function from argumentation models to theory models.

Postulate 6.1 (Closure). *Let $(Lit, StrInf, DefInf)$ be a defeasible theory. Its translation satisfies* closure *for semantics σ iff for any σ-model M, we find that $Cl_{StrInf}(Lit \cap M) \subseteq Lit \cap M$.*

Naturally, the notion of consistency is reduced to consistency of a set of literals of the underlying logical language. Note that consistency only concerns the local consistency of a given single model of the target formalism. It may well be that the formalism is globally inconsistent in the sense of not allowing for any model with respect to a particular semantics. The latter behaviour can be desired, for example if the original theory base is inconsistent already.

Postulate 6.2 (Direct Consistency). *Let $(Lit, StrInf, DefInf)$ be a defeasible theory with translation X and σ a semantics. X satisfies* direct consistency *iff for all σ-models M we have that $Lit \cap M$ is consistent.*

Caminada and Amgoud (2007) remark that it is usually easy to satisfy direct consistency, but much harder to satisfy the stronger notion of indirect consistency. For this to hold, for each model its closure under strict rules must be consistent.

Postulate 6.3 (Indirect Consistency). *Let $(Lit, StrInf, DefInf)$ be a defeasible theory with translation X and σ a semantics. X satisfies* indirect consistency *iff for all σ-models M we have that $Cl_{StrInf}(Lit \cap M)$ is consistent.*

As a counterpart to Proposition 7 of Caminada and Amgoud (2007), we can show that closure and direct consistency together imply indirect consistency.

Proposition 6.1. *Let $(Lit, StrInf, DefInf)$ be a defeasible theory with translation X and σ a semantics. If X satisfies closure and direct consistency, then it satisfies indirect consistency.*

Proof. Let X satisfy closure and direct consistency, and let M be a σ-model for X. We have to show that $Cl_{StrInf}(Lit \cap M)$ is consistent. Since X satisfies closure, $Cl_{StrInf}(Lit \cap M) \subseteq Lit \cap M$. Now since X satisfies direct consistency, $Lit \cap M$ is consistent. Hence its subset $Cl_{StrInf}(Lit \cap M) \subseteq Lit$ is consistent and X satisfies indirect consistency. □

While Caminada and Amgoud (2007) observed problematic issues in giving argument-based semantics to defeasible theory bases, they still succeeded in devising an approach that is able to achieve closure and direct and indirect consistency for any admissibility-based semantics by using appropriate (semantics-dependent) definitions of rebut and undercut.

6.2.2 The Approach of Wyner, Bench-Capon, and Dunne (2013)

Wyner et al. (2009, 2013) identified some problems of the approach of Caminada and Amgoud (2007) and proposed an alternative translation from theory bases to argumentation frameworks. We do not necessarily support or reject their philosophical criticisms, but rather find the translation technically appealing. They create an argument for each literal in the theory base's language and additionally an argument for each rule. Intuitively, the literal arguments indicate that the literal holds, and the rule arguments indicate that the rule is applicable. Furthermore, the defined conflicts between these arguments are straightforward:

1. opposite literals attack each other;
2. rules are attacked by the negations of their body literals;
3. defeasible rules are attacked by the negation of their head;

4. all rules attack the negation of their head.

Definition 6.1 (Wyner et al., 2009, Definitions 4 and 5). Let $TB = (Lit, StrInf, DefInf)$ be a defeasible theory. Define an argumentation framework $F(TB) = (A, R)$ by

$$\begin{aligned} A = \quad & Lit \cup \{r \mid r : \phi_1, \ldots, \phi_n \Rrightarrow \psi \in StrInf \cup DefInf\} \\ R = \quad & \{(\psi, \overline{\psi}) \mid \psi \in Lit\} \\ & \cup \{(\overline{\phi_i}, r) \mid r : \phi_1, \ldots, \phi_n \Rrightarrow \psi \in StrInf \cup DefInf, 1 \leq i \leq n\} \\ & \cup \{(\overline{\psi}, r) \mid r : \phi_1, \ldots, \phi_n \Rightarrow \psi \in DefInf\} \\ & \cup \{(r, \overline{\psi}) \mid r : \phi_1, \ldots, \phi_n \Rrightarrow \psi \in StrInf \cup DefInf\} \end{aligned}$$ ◇

As mentioned in the introduction to this chapter, a further definition is needed to rule out extensions that are not closed under strict rules.

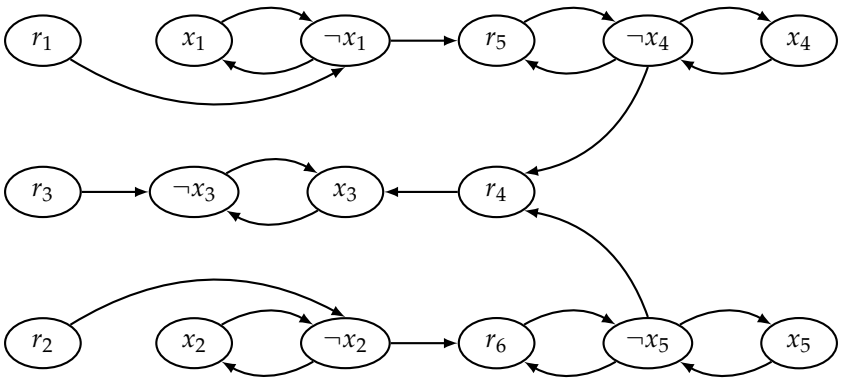
Definition 6.2 (Wyner et al., 2013, Definition 7). Let $TB = (Lit, StrInf, DefInf)$ be a defeasible theory and $F(TB) = (A, R)$ its associated argumentation framework. An extension $M \subseteq A$ of $F(TB)$ is *well-formed* if there is no strict rule $r : \phi_1, \ldots, \phi_n \to \psi \in StrInf$ such that $\{r, \phi_1, \ldots, \phi_n\} \subseteq M$ but $\psi \notin M$. ◇

It is decidable in polynomial time whether a given extension M is well-formed: we can compute $Cl_{StrInf}(M)$ and then check whether $Cl_{StrInf}(M) \subseteq M$. This means that the additional computational cost incurred by Definition 6.2 is acceptable under standard assumptions.[3] For illustration, let us now look at one of the examples of Wyner et al. (2013) which they adapted from (Caminada and Amgoud, 2007).

Example 6.2 (Wyner et al., 2013, Example 4). Consider the following theory base.

$$\begin{aligned} Lit &= \{x_1, x_2, x_3, x_4, x_5, \neg x_1, \neg x_2, \neg x_3, \neg x_4, \neg x_5\} \\ StrInf &= \{r_1 : \to x_1, \quad r_2 : \to x_2, \quad r_3 : \to x_3, \quad r_4 : x_4, x_5 \to \neg x_3\} \\ DefInf &= \{r_5 : x_1 \Rightarrow x_4, \quad r_6 : x_2 \Rightarrow x_5\} \end{aligned}$$

We can see that x_1, x_2, x_3 are strictly asserted and thus should be contained in any extension. The AF translation is depicted below.



[3] It might even be possible to encode Definition 6.2 directly into the translated AF by adding, for each strict rule $r : \phi_1, \ldots, \phi_n \to \psi \in StrInf$, a new argument -$r$ ("r is inapplicable") and the attacks (-r,-r), (ψ,-r), ($\overline{\phi_1}$,-r), …, ($\overline{\phi_n}$,-r).

The stable extensions of this AF are as follows:

$$S_1 = \{x_1, x_2, x_3, \neg x_4, \neg x_5, r_1, r_2, r_3\} \qquad S_2 = \{x_1, x_2, x_3, \neg x_4, x_5, r_1, r_2, r_3, r_6\}$$
$$S_3 = \{x_1, x_2, x_3, x_4, \neg x_5, r_1, r_2, r_3, r_5\} \qquad S_4 = \{x_1, x_2, x_4, x_5, r_1, r_2, r_3, r_4, r_5, r_6\}$$

While the first three extensions can be considered intended, S_4 is not closed under strict rules and indirectly inconsistent: r_3 is applicable but x_3 does not hold, r_4 is applicable but $\neg x_3$ does not hold. Indeed, S_4 is not well-formed and thus should not be considered for drawing conclusions (Wyner et al., 2013). ◇

A similar observation can be made in Example 6.1: the AF translation according to Wyner et al. (2009) has a stable extension $\{w, g, m, b, r_1, r_2, r_3, r_4, r_5, r_6\}$ where John is a married bachelor; but again, this extension is not well-formed and thus discarded.

6.3 Instantiations to Abstract Dialectical Frameworks

In this section, we extend the theory base to AF translation of Wyner et al. (2009) to ADFs. Due to the availability of support, this is straightforward. Indeed, support and attack are sufficient for our purposes and we can therefore restrict our attention to bipolar ADFs.

6.3.1 From Theory Bases to ADFs

As in the approach of Wyner et al. (2009), we directly use the literals from the theory base as statements that express whether the literal holds. We also use rule names as statements indicating that the rule is applicable. Additionally, for each rule r we use a statement -r indicating that the rule has not been applied. Not applying a rule is acceptable for defeasible rules, but unacceptable for strict rules since it would violate the closure postulate. This is enforced via integrity constraints saying that it may not be the case in any model that the rule body holds but the head does not hold. Technically, for a strict rule r, we introduce a conditional self-attack of -r; this self-attack becomes active if (and only if) the body of r is satisfied but the head of r is not satisfied, thereby preventing this undesirable state of affairs from getting included in a model. Defeasible rules offer some degree of choice, whence we leave it to the semantics whether or not to apply them. This choice is modelled by a mutual attack cycle between r and -r. The remaining acceptance conditions are equally straightforward:

- Opposite literals attack each other.
- A literal is accepted whenever some rule deriving it is applicable, that is, all rules with head ψ support statement ψ.
- A strict rule is applicable whenever all of its body literals hold, that is, the body literals of r are exactly the supporters of r.
- Likewise, a defeasible rule is applicable whenever all of its body literals hold, and additionally the negation of its head literal must not hold.

In particular, literals cannot be accepted unless there is some rule deriving them.

Definition 6.3. Let $TB = (Lit, StrInf, DefInf)$ be a theory base. Define an ADF $D(TB) = (S, L, C)$ by $S = Lit \cup \{r, \text{-}r \mid r : \phi_1, \ldots, \phi_n \Rrightarrow \psi \in StrInf \cup DefInf\}$; the acceptance functions of statements s can be parsimoniously represented by propositional formulas φ_s:[4]

For a literal $\psi \in Lit$, we define

$$\varphi_\psi = \neg[\overline{\psi}] \wedge \bigvee_{r:\phi_1,\ldots,\phi_n \Rrightarrow \psi \in StrInf \cup DefInf} [r]$$

For a strict rule $r : \phi_1, \ldots, \phi_n \to \psi \in StrInf$, we define

$$\varphi_r = [\phi_1] \wedge \ldots \wedge [\phi_n] \qquad \text{and}$$
$$\varphi_{\text{-}r} = [\phi_1] \wedge \ldots \wedge [\phi_n] \wedge \neg[\psi] \wedge \neg[\text{-}r]$$

For a defeasible rule $r : \phi_1, \ldots, \phi_n \Rightarrow \psi \in DefInf$, we define

$$\varphi_r = [\phi_1] \wedge \ldots \wedge [\phi_n] \wedge \neg[\overline{\psi}] \wedge \neg[\text{-}r] \qquad \text{and}$$
$$\varphi_{\text{-}r} = \neg[r]$$

Finally, there is a link $(s', s) \in L$ iff $[s']$ occurs in the acceptance formula φ_s. ◇

(For the formulas defined above, the empty disjunction leads to $\bot$ – logical falsity – and the empty conjunction to $\top$ – logical truth.)

Let us see how our translation treats the examples seen earlier.

Example 6.3 (Continued from Example 6.2). Definition 6.3 yields these acceptance formulas:

$$\begin{aligned}
\varphi_{x_1} &= \neg[\neg x_1] \wedge [r_1] & \varphi_{\neg x_1} &= \bot \\
\varphi_{x_2} &= \neg[\neg x_2] \wedge [r_2] & \varphi_{\neg x_2} &= \bot \\
\varphi_{x_3} &= \neg[\neg x_3] \wedge [r_3] & \varphi_{\neg x_3} &= \neg[x_3] \wedge [r_4] \\
\varphi_{x_4} &= \neg[\neg x_4] \wedge [r_5] & \varphi_{\neg x_4} &= \bot \\
\varphi_{x_5} &= \neg[\neg x_5] \wedge [r_6] & \varphi_{\neg x_5} &= \bot \\
\varphi_{r_1} &= \top & \varphi_{\text{-}r_1} &= \neg[x_1] \wedge \neg[\text{-}r_1] \\
\varphi_{r_2} &= \top & \varphi_{\text{-}r_2} &= \neg[x_2] \wedge \neg[\text{-}r_2] \\
\varphi_{r_3} &= \top & \varphi_{\text{-}r_3} &= \neg[x_3] \wedge \neg[\text{-}r_3] \\
\varphi_{r_4} &= [x_4] \wedge [x_5] & \varphi_{\text{-}r_4} &= [x_4] \wedge [x_5] \wedge \neg[\neg x_3] \wedge \neg[\text{-}r_4] \\
\varphi_{r_5} &= [x_1] \wedge \neg[\neg x_4] \wedge \neg[\text{-}r_5] & \varphi_{\text{-}r_5} &= \neg[r_5] \\
\varphi_{r_6} &= [x_2] \wedge \neg[\neg x_5] \wedge \neg[\text{-}r_6] & \varphi_{\text{-}r_6} &= \neg[r_6]
\end{aligned}$$

Statements with an acceptance condition of the form $\neg p_1 \wedge \ldots \wedge \neg p_n$ behave like AF arguments. So in particular r_1, r_2, r_3 are always **t** since these rules have an empty body. Similarly, $\text{-}r_1, \text{-}r_2, \text{-}r_3$ are self-attacking arguments. The statements $\neg x_1, \neg x_2, \neg x_4, \neg x_5$ are always **f** since there are no rules deriving these literals. The remaining acceptance conditions are clear from the definitions: literals are supported by the rules deriving them and rules in turn are supported by their body literals.

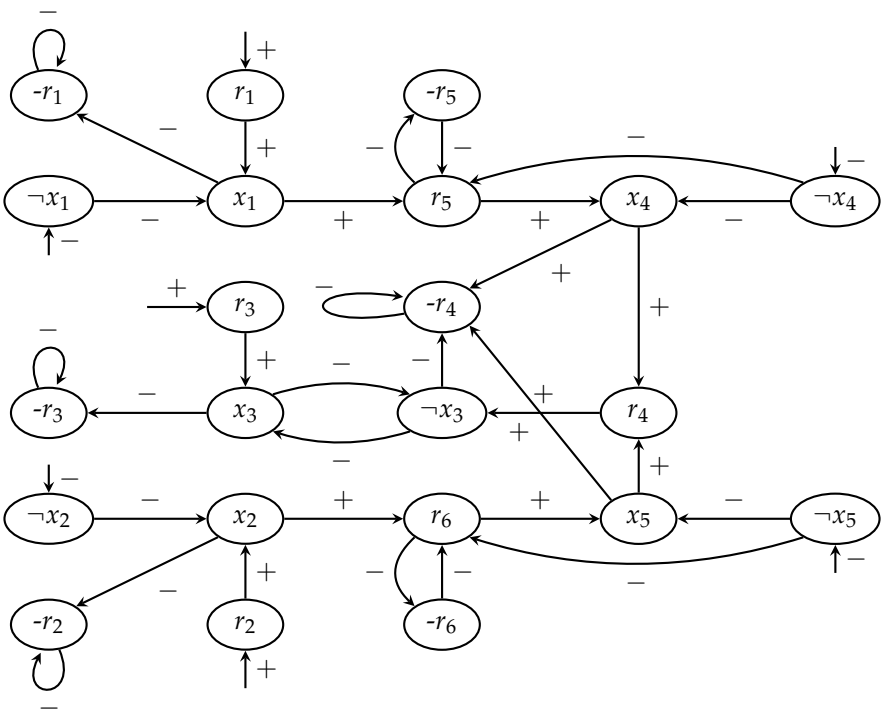


Figure 6.1: Graphical depiction of the ADF from Example 6.2. Edge labels + and − indicate supporting and attacking links. Several statements have constant truth values as acceptance conditions, in the picture this is indicated via a link from the surroundings. (This is inspired by conventions from automata theory, where initial states are indicated likewise.)

For illustration, we also provide the ADF in form of a labelled graph in Figure 6.1. For this ADF, models and stable models coincide, and there are three of them:

$$M_1 = \{x_1, x_2, x_3, r_1, r_2, r_3, \text{-}r_5, \text{-}r_6\} \qquad M_2 = \{x_1, x_2, x_3, x_4, r_1, r_2, r_3, r_5, \text{-}r_6\}$$
$$M_3 = \{x_1, x_2, x_3, x_5, r_1, r_2, r_3, \text{-}r_5, r_6\}$$

Roughly, in M_1 none of the defeasible rules r_5, r_6 has been applied – indicated by -r_5 and -r_6 –, while in M_2 and M_3 either one of them has been applied. As intended, there is no model where both defeasible rules have been applied, as this would lead to a set that contains both x_4 and x_5; this in turn would make rule r_4 applicable, allowing to conclude $\neg x_3$ in contradiction to x_3 being strictly true according to rule r_3. We can furthermore see that all of the models are closed under strict rule application (they contain x_1, x_2, x_3 and no other strict rule is applicable) and directly consistent, thus also indirectly consistent. ◇

A similar observation can be made for John (not) being married (Example 6.1); our ADF translation has three (stable) models: $M_1 = \{w, g, r_1, r_2, \text{-}r_5, \text{-}r_6\}$,

[4] In these formulas, we write ADF statements in brackets, to avoid confusion between negation being applied inside a statement name – as in $[\neg x]$ – and negation being applied in the formula outside of the statement's name – as in $\neg[\text{-}r]$. Thus $[\neg x]$ and $\neg[x]$ are syntactically different literals in the language of acceptance formulas; their meaning is intertwined via the semantics of ADFs.

$M_2 = \{w, g, h, m, r_1, r_2, r_4, r_5, \text{-}r_6\}$ and $M_3 = \{w, g, b, \neg h, r_1, r_2, r_3, \text{-}r_5, r_6\}$. Again, the argumentation translation of the theory base satisfies closure and direct and indirect consistency. We will later prove that the satisfaction of the postulates is not a coincidence in our approach. But first of all let us consider another problem which often arises in knowledge representation and reasoning.

6.3.2 Support Cycles in Theory Bases

When logical, rule-based approaches are used for knowledge representation, a recurring issue is that of cyclic dependencies between propositions of the knowledge base. If such support cycles are carelessly overlooked or otherwise not treated in an adequate way, they can lead to counterintuitive conclusions. Consider this famous example by Denecker, Theseider-Dupré, and Van Belleghem (1998).

Example 6.4 (Gear Wheels; Denecker et al., 1998). There are two interlocked gear wheels x and y that can be separately turned and stopped. Let x_0 and y_0 denote whether x (resp. y) turns at time point 0, and likewise for a successive time point 1. At any one time point, whenever the first wheel turns (resp. stops), it causes the second one to turn (resp. stop), and vice versa. This is expressed by strict rules r_1 to r_8. Without a cause for change, things usually stay the way they are from one time point to the next, which is expressed by the defeasible rules r_a to r_d.

$$
\begin{aligned}
\mathit{Lit} &= \{x_0, y_0, x_1, y_1, \neg x_0, \neg y_0, \neg x_1, \neg y_1\} \\
\mathit{StrInf} &= \{r_1 : x_0 \to y_0, \quad r_2 : y_0 \to x_0, \quad r_3 : \neg x_0 \to \neg y_0, \quad r_4 : \neg y_0 \to \neg x_0, \\
&\qquad r_5 : x_1 \to y_1, \quad r_6 : y_1 \to x_1, \quad r_7 : \neg x_1 \to \neg y_1, \quad r_8 : \neg y_1 \to \neg x_1\} \\
\mathit{DefInf} &= \{r_a : x_0 \Rightarrow x_1, \quad r_b : \neg x_0 \Rightarrow \neg x_1, \quad r_c : y_0 \Rightarrow y_1, \quad r_d : \neg y_0 \Rightarrow \neg y_1\}
\end{aligned}
$$

For later reference, we denote this theory base by $TB_{GW} = (\mathit{Lit}, \mathit{StrInf}, \mathit{DefInf})$. To model a concrete scenario, we add the rules $\mathit{StrInf}' = \{r_i :\to \neg x_0, r_j :\to \neg y_0\}$ expressing that both wheels initially stand still. We denote the augmented theory base for this concrete scenario by $TB'_{GW} = (\mathit{Lit}, \mathit{StrInf} \cup \mathit{StrInf}', \mathit{DefInf})$. It is clearly unintended that there is some model for TB'_{GW} where the gear wheels magically start turning with one being the cause for the other and vice versa. ◇

The same holds for defeasible rules.

Example 6.5 (Defeasible cycle). Consider these defeasible rules saying that *rain* and *wet* grass usually go hand in hand: $\mathit{Lit} = \{\mathit{rain}, \mathit{wet}, \neg\mathit{rain}, \neg\mathit{wet}\}$, $\mathit{StrInf} = \emptyset$ and $\mathit{DefInf} = \{r_1 : \mathit{rain} \Rightarrow \mathit{wet}, r_2 : \mathit{wet} \Rightarrow \mathit{rain}\}$. The intended meaning is that one is usually accompanied by the other, not that both may appear out of thin air. ◇

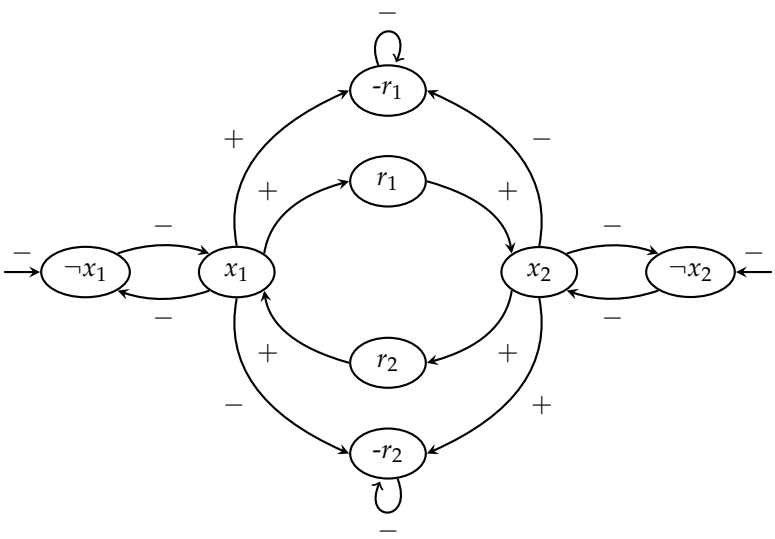
To see how argumentation translations of theory bases treat such cycles, let us look at a simplified version of the gear wheels example.

Example 6.6 (Strict cycle). Consider a theory base with two literals mutually supporting each other through strict rules: $\mathit{Lit} = \{x_1, x_2, \neg x_1, \neg x_2\}$, the strict rules are given by

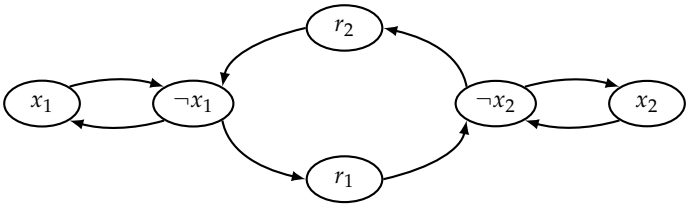
$StrInf = \{r_1 : x_1 \to x_2, \quad r_2 : x_2 \to x_1\}$ and $DefInf = \emptyset$. Our ADF translation of this example yields the acceptance formulas

$$\varphi_{x_1} = [r_2] \qquad \varphi_{\neg x_1} = \bot \qquad \varphi_{r_1} = [x_1] \qquad \varphi_{\text{-}r_1} = [x_1] \wedge \neg[x_2] \wedge \neg[\text{-}r_1]$$
$$\varphi_{x_2} = [r_1] \qquad \varphi_{\neg x_2} = \bot \qquad \varphi_{r_2} = [x_2] \qquad \varphi_{\text{-}r_2} = [x_2] \wedge \neg[x_1] \wedge \neg[\text{-}r_2]$$

We can see that the support cycle between x_1 and x_2 in *StrInf* is directly translated into a support cycle between the ADF statements x_1, r_1, x_2, r_2 in the centre of the picture.



The ADF has two models, $M_1 = \{x_1, x_2, r_1, r_2\}$ and $M_2 = \emptyset$. Only M_2 is a stable model due to the cyclic self-support of the statements in M_1. Note that not only do x_1 and x_2 not hold in M_2, neither do $\neg x_1$ and $\neg x_2$ (there are no rules possibly deriving them). In contrast, the translation of Wyner et al. (2009) yields the AF



with two stable extensions $S_1 = \{x_1, r_1, x_2, r_2\}$ and $S_2 = \{\neg x_1, \neg x_2\}$. In S_1, x_1 and x_2 hold due to self-support while in S_2 they are "guessed" to be false.[5] ◇

In our view, this is problematic since it is not made clear to the user that these different extensions arise due to self-support. Even if we grant that for some application domains, cyclic self-support of literals might be intended or at least not unintended, the user should be able to distinguish whether different models/extensions arise due to present or absent self-support on the one hand, or due to conflicts between defeasible conclusions on the other hand. ADFs provide this important distinction, since cycles are allowed in models and disallowed in stable

[5] In this chapter, we consider only stable extension semantics for AFs. It might be possible to choose/come up with an AF semantics that treats the above AF differently.

models, while both semantics are identical in their treatment of conflicts between defeasible conclusions.

In the approach of Caminada and Amgoud (2007), treatment of cycles is built into the definition of the set of arguments in the resulting argumentation framework. The arguments are created using structural induction, where rules with empty bodies form the induction base and all other rules form the induction step. For the general gear wheel domain TB_{GW} of Example 6.4, and for Examples 6.5 and 6.6, their translation would not create any arguments (there are no assertions in the theory bases), and the approach could not draw any conclusions about these examples. The concrete scenario of the interlocked gear wheel domain TB'_{GW} in Example 6.4, where both wheels initially stand still, would be treated correctly by the approach of Caminada and Amgoud (2007). But note that the well-foundedness of the treatment of cyclic dependencies is built into the syntax of the resulting argumentation framework – there are no arguments that could conclude that any of the wheels is turning, although there are (strict and defeasible) rules with such conclusions.[6] Consequently, a part of the semantics of the theory base is already fixed by the translation, irrespective of the argumentation semantics that is used later on.

6.3.3 Inconsistent Theory Bases

Example 6.7 (Inconsistent Theory Base). Consider the following (obviously inconsistent) theory base in which both a literal and its negation are strictly asserted: $Lit = \{x, \neg x\}$, $StrInf = \{r_1 :\rightarrow x, \quad r_2 :\rightarrow \neg x\}$ and $DefInf = \emptyset$. Our ADF translation yields the acceptance formulas

$$\varphi_x = \neg[\neg x] \wedge [r_1] \qquad \varphi_{r_1} = \top \qquad \varphi_{\text{-}r_1} = \neg[x] \wedge \neg[\text{-}r_1]$$
$$\varphi_{\neg x} = \neg[x] \wedge [r_2] \qquad \varphi_{r_2} = \top \qquad \varphi_{\text{-}r_2} = \neg[\neg x] \wedge \neg[\text{-}r_2]$$

This ADF has no models, and so the theory base's inconsistency is detected.

On the other hand, the associated argumentation framework due to Wyner et al. (2009) is given by the set of arguments $A = \{x, \neg x, r_1, r_2\}$ and the attacks $R = \{(x, \neg x), (\neg x, x), (r_1, \neg x), (r_2, x)\}$. In the only stable extension $\{r_1, r_2\}$ both rules are applicable but none of the head literals hold due to immanent conflict. Again, this extension is not well-formed and the inconsistency is made obvious.

In the approach of Caminada and Amgoud (2007), we can construct two strict arguments that conclude x and $\neg x$, respectively. There are no attacks between these arguments, since rebuts are impossible between strict arguments and rules without body cannot be undercut. So their resulting AF has a stable extension from which both x and $\neg x$ can be concluded, which detects the inconsistency. ◇

6.3.4 Properties of the Translation

In this section, we analyse some theoretical properties of our translation. First we show that it satisfies (our reformulations of) the rationality postulates of Caminada and Amgoud (2007). Then we analyse the computational complexity of translating a given theory base and show that the blowup is at most quadratic.

[6] See also the discussion of the (non-)treatment of *partial* knowledge bases by Wyner et al. (2013).

Postulates It is elementary to show that the ADFs resulting from our translation satisfy direct consistency. This is because the statements ψ and $\overline{\psi}$ mutually attack each other.

Proposition 6.2. *For any theory base $TB = (Lit, StrInf, DefInf)$, its associated ADF $D(TB)$ satisfies direct consistency with respect to the model semantics.*

Proof. Let M be a model for $D(TB)$ and assume to the contrary that $M \cap Lit$ is inconsistent. Then there is a $\psi \in Lit$ such that $\psi \in M$ and $\neg\psi \in M$. Since $\neg\psi \in M$, the acceptance condition of $\neg\psi$ yields $\psi \notin M$. Contradiction. □

We can also prove that they satisfy closure: by construction, the (acceptance conditions of) statements *-r* for strict rules *r* guarantee that the rule head is contained in any model that contains the rule body.

Proposition 6.3. *For any theory base $TB = (Lit, StrInf, DefInf)$, its associated ADF $D(TB)$ satisfies closure with respect to the model semantics.*

Proof. Let M be a model of $D(TB)$ and $r : \phi_1, \ldots, \phi_n \to \psi \in StrInf$ such that we find $\phi_1, \ldots, \phi_n \in M$. We have to show $\psi \in M$. By definition, $D(TB)$ has a statement -r with parents $par(\text{-}r) = \{\phi_1, \ldots, \phi_n, \psi, \text{-}r\}$. We next show that -r $\notin M$: assume to the contrary that -r $\in M$. Then by the acceptance condition of -r we get -r $\notin M$, contradiction. Thus -r $\notin M$. Now the acceptance condition of -r yields $\phi_1 \notin M$ or ... or $\phi_n \notin M$ or $\psi \in M$ or -r $\in M$. By assumption, we have $\phi_1, \ldots, \phi_n \in M$ and -r $\notin M$, thus we get $\psi \in M$. □

By Proposition 6.1 the translation satisfies indirect consistency.

Corollary 6.4. *For any theory base $TB = (Lit, StrInf, DefInf)$, its associated ADF $D(TB)$ satisfies indirect consistency with respect to the model semantics.*

Since any stable model is a model, our translation also satisfies the postulates for the stable model semantics.

Corollary 6.5. *For any theory base $TB = (Lit, StrInf, DefInf)$, its associated ADF $D(TB)$ satisfies closure and direct and indirect consistency with respect to the stable model semantics.*

It should be noted that defeasible rules may or may not be applied – the approach is not "eager" to apply defeasible rules.

Complexity For a theory base $TB = (Lit, StrInf, DefInf)$, we define the size of its constituents as follows. Quite straightforwardly, the size of a set of literals is just its cardinality, the size of a rule is the number of literals in it, the size of a set of rules is the sum of the sizes of its elements and the size of a theory base is the sum of the sizes of its components.

$$\begin{aligned} \|Lit\| &= |Lit| \\ \|r : \phi_1, \ldots, \phi_n \Rrightarrow \psi\| &= n + 1 \\ \|StrInf\| &= \sum_{r \in StrInf} \|r\| \\ \|DefInf\| &= \sum_{r \in DefInf} \|r\| \\ \|TB\| &= \|Lit\| + \|StrInf\| + \|DefInf\| \end{aligned}$$

We want to analyse the size of its ADF translation $D(TB) = (S, L, C)$ according to Definition 6.3. Clearly, the number of statements is linear in the size of the theory base, since we have one statement for each literal and two statements for each rule: $|S| = |Lit| + 2 \cdot (|StrInf| + |DefInf|)$. Since $L \subseteq S \times S$, the number of links in L is at most quadratic in the cardinality of S: $|L| \leq |S|^2$. Finally, we have seen in Definition 6.3 that the acceptance conditions of statements can be parsimoniously represented by propositional formulas. It can be checked that the size of each one of these formulas is at most linear in the size of the theory base. Since there are linearly many statements with one acceptance formula each, the acceptance conditions can be represented in quadratic space. So overall, the resulting ADF $D(TB) = (S, L, C)$ can be represented in space which is at most quadratic in the size of the original theory base. In particular, in our approach a finite theory base always yields a finite argumentation translation. This is in contrast to the definition of Caminada and Amgoud (2007), where the strict rule set $StrInf = \{r_0 :\rightarrow a, \quad r_1 : a \rightarrow b, \quad r_2 : b \rightarrow a\}$ allows to construct infinitely many arguments $A_1 = [\rightarrow a], A_2 = [A_1 \rightarrow b], A_3 = [A_2 \rightarrow a], A_4 = [A_3 \rightarrow b], \ldots$[7]

6.4 A Direct Semantics for Defeasible Theory Bases

We have previously seen how ADFs can be used to give a semantics to defeasible theory bases. Albeit we introduced additional, merely technical statements (like -r), we were able to address shortcomings of previous approaches. Still, there remains the issue that the ADF-based semantics is not necessarily eager to apply defeasible rules. In what follows, we will introduce a direct semantics for defeasible theory bases that possesses this eagerness property. It will additionally allow us to more precisely clarify our intuitions about what rules mean, especially the difference between strict and defeasible rules. While our intuitions on defeasible rules are quite clear, we will argue that there are two different intuitions on strict rules. One intuition says that strict rules are directed inference rules that operate on the knowledge level, that is, whenever the premises are known then the conclusion is inferred. In particular, in being directed these rules do not automatically entail any of their contrapositives. Let us call this intuition (DR) for *directed inference rule*; we will see that (DR) can lead to problems with global inconsistency. Another intuition says that strict rules are just like material implications in propositional logic, let us call it (MI). In particular, in this intuition strict rules are not directed and therefore equivalent to their contrapositives.[8] (MI) is unproblematic in its interaction with defeasible rules, but raises the philosophical question why strict rules should allow for contraposition and defeasible rules should not. These questions are pervasive in giving semantics to nonmonotonic rule-based systems, and may account for (parts of) the complications encountered by Caminada and Amgoud (2007).

To formalise the two mentioned intuitions, we make use of concepts from epistemic modal logic. We consider epistemic states in the form of sets of possible worlds, where a possible world is simply a two-valued interpretation of a propositional vocabulary. More precisely, let A be a propositional signature. Then an interpretation over A can be represented as a set $w \subseteq A$ as usual; we will also call an interpretation a *world*. We then define the *set of worlds over* A as $W_A = 2^A$. A set $Q \subseteq W_A$ is then an *epistemic state*: intuitively, any entity being in the epistemic state Q considers all and only the worlds $w \in Q$ to be possible, that is, to be the one single "real" world the entity "lives in." Put another way, an epistemic state Q signifies that

[7] Even if we exclude cycles in rules, there are rule sets that allow for exponentially many arguments: Set $D_0 = \{\Rightarrow p_0, \Rightarrow \neg p_0\}$, $D_1 = D_0 \cup \{p_0 \Rightarrow p_1, \neg p_0 \Rightarrow p_1\}$ and for $i \geq 1$, $D_{i+1} = D_i \cup \{p_0, p_i \Rightarrow p_{i+1}, \neg p_0, p_i \Rightarrow p_{i+1}\}$. For any $n \in \mathbb{N}$, the size of D_n is linear in n and D_n leads to 2^{n+1} arguments, among them 2^n arguments for p_n.

[8] Caminada and Amgoud have a similar concept, *closure under transposition* (Caminada and Amgoud, 2007, Def. 17).

any entity subscribing to this epistemic state cannot distinguish the worlds in Q with what it knows. (But it can distinguish worlds *in* Q from those *not in* Q.) The knowledge associated with an epistemic state Q over A is simply the set of propositional formulas over A which are true in all possible worlds, the theory $\{\varphi \mid w \models \varphi \text{ for all } w \in Q\}$.

We start to formalise the intuition (MI), where strict rules $\phi_1, \ldots, \phi_n \to \psi$ are interpreted as material implications $(\phi_1 \wedge \ldots \wedge \phi_n) \to \psi$ in propositional logic. To do this, we define a satisfaction relation $\models$, that indicates whether an epistemic state together with a specific world (the "real world") satisfies an element of a defeasible theory base. Of course, it is trivial to define this for literals. For strict rules, the real world must satisfy the above material implication. For defeasible rules $r : \phi_1, \ldots, \phi_n \Rightarrow \psi$, our intuition is as follows: Assume that w is the real world and Q is our epistemic state. If we *know* that all body literals $\phi_1, \ldots, \phi_n$ hold, and we do *not* know that the conclusion is false, then for the pair Q, w to satisfy the defeasible rule, the conclusion must hold in the real world w. Otherwise, quite simply, the defeasible rule would not be a very valuable guide on what normally holds in the world. In our formalisation below, this intuition is split up into three ways how a defeasible rule can be satisfied:

1. Not all of the body literals are known. (Then the rule is inapplicable due to insufficient premises.)
2. The negation of the head literal is known. (Then the rule is inapplicable due to an exception.)
3. The head literal is actually true. (Then the defeasible rule is good regardless of what we know, because it tells us something true about the world.)

Definition 6.4. Let $TB = (Lit, StrInf, DefInf)$ be a defeasible theory. Let $A \subseteq Lit$ be all atoms of the language, $a \in A$, $w \in W_A$ and $Q \subseteq W_A$.[9]

$$
\begin{array}{rl}
w \models a & \text{iff } a \in w \\
w \models \neg a & \text{iff } a \notin w \\
w \models r : \phi_1, \ldots, \phi_n \to \psi & \text{iff } w \models (\phi_1 \wedge \ldots \wedge \phi_n) \to \psi \text{ in propositional logic} \\
Q, w \models r : \phi_1, \ldots, \phi_n \Rightarrow \psi & \text{iff there is a } v \in Q \text{ and } 1 \leq i \leq n \text{ with } v \not\models \phi_i \\
 & \quad \text{or for all } v \in Q \text{ we have } v \models \psi \\
 & \quad \text{or } w \models \psi \\
Q, w \models TB & \text{iff } Q, w \models r \text{ for all } r \in StrInf \cup DefInf
\end{array}
$$ ◇

We use Examples 1 and 2 from the work of Wyner et al. (2013) to illustrate the definitions.

Example 6.8 (Partial theories). Consider the set of literals $Lit = \{x_1, x_2, \neg x_1, \neg x_2\}$; then the set of atoms is $A = \{x_1, x_2\}$. Consequently, there are four possible worlds, that is, $W_A = \{\emptyset, \{x_1\}, \{x_2\}, \{x_1, x_2\}\}$. It follows that 2^{W_A} contains $2^4 = 16$ different epistemic states, among them the state W_A where any world is considered possible (thus the agent knows nothing) and the state $\emptyset$ where the agent's knowledge is inconsistent.

Considering the strict rule $r_1 : x_1 \to x_2$, it is easy to see that it is satisfied by all worlds except $\{x_1\}$. For its defeasible variant $r_2 : x_1 \Rightarrow x_2$ we have the following: Assume the epistemic state $Q = \{\{x_1\}, \{x_1, x_2\}\}$ where we know that x_1 is true but are oblivious whether x_2 holds, and the real world $w = \{x_1\}$ where x_2 is false. Then we have $Q, w \not\models r_2 : x_1 \Rightarrow x_2$ since we know that the rule's body is true, do not know that its head is false, but its head is false in the real world. For $w' = \{x_1, x_2\}$, we would get $Q, w' \models r_2 : x_1 \Rightarrow x_2$ since $w' \models x_2$. ◇

[9] For conciseness, we leave out the epistemic state or the real world when it is not used in the definition of the satisfaction relation.

With the satisfaction relation at hand, it is then straightforward to define when an epistemic state Q is a model of a defeasible theory: whenever Q coincides with the set of possible worlds w for which the pair Q, w satisfies all rules in the defeasible theory base.

Definition 6.5. For a theory base TB, a set $Q \subseteq W_A$ of possible worlds is a *model for TB* if and only if $Q = \{w \in W_A \mid Q, w \models TB\}$. ◇

Example 6.9 (Continued from Example 6.8). For the defeasible theory base TB_1 consisting only of the strict rule $r_1 : x_1 \to x_2$, we get a single model $Q_1 = \{\emptyset, \{x_2\}, \{x_1, x_2\}\}$. In Q_1 we know that x_1 implies x_2, but we do not know anything else. These possible worlds correspond one-to-one with the preferred extensions that Wyner et al. obtain for the very same theory (Wyner et al., 2013, Example 1).

For the defeasible theory base TB_2 consisting only of the defeasible rule $r_2 : x_1 \Rightarrow x_2$, the only model is $Q_2 = W_A$ where all worlds are considered possible. Intuitively, the premise of the defeasible rule is not known, and so the rule cannot be applied. ◇

Let us consider some further examples.

Example 6.10 (Continued from Example 6.1). For the married John example, we get two models:

$$Q_1 = \{\{g, w, b\}\} \quad \text{and} \quad Q_2 = \{\{g, w, m, h\}\}$$

In both models, the epistemic state is fully determined, that is, we know exactly which world is the real one. In Q_1, John is a bachelor; in Q_2, he is married and thus has a spouse. In both epistemic states, John goes out and wears a ring. Note that the semantics is eager to apply defeasible rules – while Q_1 and Q_2 directly correspond to the models M_3 and M_2 (page 137) of the ADF translation, there is no possible-worlds equivalent of M_1 where no defeasible rule has been applied. The reason for this is easy to see: if the epistemic state $Q_3 = \{\{g, w\}\}$ were a model, then we would have $Q_3 = \{w \in W_A \mid Q_3, w \models TB\}$. However for $v' = \{g, w, b\}$, we find that $Q_3, v' \models TB$ but $v' \notin Q_3$. Intuitively, the pair Q_3, v' satisfies the defeasible rule $r_6 : g \Rightarrow b$ because $v' \models b$; so according to what is known v' should be considered a possible world, but Q_3 does not do so. (The same can be shown for $v'' = \{g, w, m, h\}$.) ◇

The behaviour of the other problematic example follows suit.

Example 6.11 (Continued from Example 6.3). Again, we get two models:

$$Q_1 = \{\{x_1, x_2, x_3, x_4\}\} \quad \text{and} \quad Q_2 = \{\{x_1, x_2, x_3, x_5\}\}$$

In both models, the set of applicable (and applied) defeasible rules is maximal; in contrast to the AF- and ADF-based semantics, there is no third model in which no defeasible rule has been applied. ◇

We consider this eagerness to apply defeasible rules one of the most important differences between our direct semantics and the several previously seen translation-based semantics. As another difference, the outcome (model) of the possible-worlds semantics is not a propositional valuation, but a propositional theory (the set of all propositional formulas that are true in all worlds that are considered possible by the epistemic state). With respect to consistency of this theory, we note that, given a defeasible theory base TB, there are essentially two possibilities for its possible-worlds semantics:

1. TB has the empty epistemic state as its only model;

2. *TB* has a non-empty model.

The first case is an indication of inconsistency on the level of strict rules.

Example 6.12 (Continued from Example 6.7). Recall the defeasible theory base comprising $Lit = \{x, \neg x\}$, $StrInf = \{r_1 :\rightarrow x,\ r_2 :\rightarrow \neg x\}$ and $DefInf = \emptyset$. We see that none of the possible worlds $\emptyset$ and $\{x\}$ satisfies *both* strict rules. Thus the rule base has the model $\emptyset$ where no world is possible and its inconsistency is obviated. ◇

For each model of a defeasible theory base, we have by definition that all its possible worlds satisfy the material implications associated with the strict rules. Thus, closure holds in each possible world and in particular in the propositional theory derived from the epistemic state.

So consistency and closure do not pose problems for the possible-worlds semantics. However, it has its issues with positive cyclic dependencies.

Example 6.13 (Continued from Example 6.5). Recall the example saying that rain and wet grass usually accompany each other: $DefInf = \{r_1 : rain \Rightarrow wet, r_2 : wet \Rightarrow rain\}$. The theory base has two models, $Q_1 = W_A$ and $Q_2 = \{\{rain, wet\}\}$. In Q_1 nothing is known about rain or wet grass; in Q_2 both are known, where each is defeasibly derived from the other. ◇

Such issues, which are problematic with regard to causality, motivate us to define a refined version of the model semantics that excludes such cycles, a *stable model semantics*. Roughly, for a model to be stable, there must be a constructive derivation of its defeasible conclusions. For instance, Q_2 above is not stable since the two conclusions cyclically depend on each other.

To achieve this constructiveness technically, we need a refinement of the satisfaction relation for defeasible rules and the notion of a model. The key change is not to check satisfaction of a rule's body against the model itself, but to check that all defeasible conclusions can be derived either from strict knowledge or from defeasible conclusions that are themselves constructively derived. This intuition comes from similar constructions in logic programming and default logic. The more technical description is to try to reconstruct a given model in an acyclic way. This construction starts with the set W_A of all possible worlds. There, nothing is known because any world is considered possible. The construction now step-wise removes worlds that are no longer considered possible. The worlds violating some strict rules are the first to go. If this leads to an increase in knowledge, then defeasible rules might become applicable and are applied through the refined model relation. If this leads to a further increase in knowledge (that is, a further decrease in the set of possible worlds), then the process continues. Otherwise the process stops, in which case we check what has been constructed. If the model could be fully reconstructed, then it is stable, otherwise it is not.

Definition 6.6. Let *TB* be a defeasible theory base over a vocabulary A, $w \in W_A$ be a world and $Q, R \subseteq W_A$ be epistemic states.

$$
\begin{aligned}
Q, R, w &\models r : \phi_1, \ldots, \phi_n \rightarrow \psi \text{ iff } w \models (\phi_1 \wedge \ldots \wedge \phi_n) \rightarrow \psi \text{ in propositional logic} \\
Q, R, w &\models r : \phi_1, \ldots, \phi_n \Rightarrow \psi \text{ iff there is a } v \in R \text{ and } 1 \leq i \leq n \text{ with } v \not\models \phi_i \\
&\qquad\qquad \text{or for all } v \in Q \text{ we have } v \not\models \psi \\
&\qquad\qquad \text{or } w \models \psi \\
Q, R, w &\models TB \text{ iff } Q, R, w \models r \text{ for all } r \in StrInf \cup DefInf
\end{aligned}
$$

Now set $R_0 = W_A$ and for $i \geq 0$ define

$$R_{i+1} = \{w \in W_A \mid Q, R_i, w \models TB\} \quad \text{and} \quad R_\infty = \bigcap_{i \geq 0} R_i$$

A set Q of possible worlds is a *stable model for TB* iff $Q = R_\infty$. ◇

It can be shown that the name "stable *model*" is well-chosen in that every stable model is a model (Denecker et al., 2003).[10]

Example 6.14 (Continued from Example 6.13). Let us check if $Q_2 = \{\{rain, wet\}\}$ is stable. We initialise the set of possible worlds $R_0 = W_A = \{\emptyset, \{rain\}, \{wet\}, \{rain, wet\}\}$. Now for obtaining R_1 according to Definition 6.6, we observe that neither defeasible rule's premise is known in the epistemic state R_0 and we have $Q_2, R_0, w \models TB$ for every world $w \in W_A$. Thus $R_1 = R_0 = W_A$, we could not reconstruct Q_2 and therefore it is not a stable model.

For $Q_1 = W_A$, on the other hand, the process terminates likewise after the first step. In this case, Q_1 could be reconstructed and is thus stable. ◇

While the stable model semantics can deal with defeasible cycles, it is at a loss with respect to strict cycles, that is, positive cyclic dependencies among literals in strict rules.

Example 6.15 (Continued from Example 6.6). Recall that the only rules of this example are strict, and given by $StrInf = \{r_1 : x_1 \to x_2, \quad r_2 : x_2 \to x_1\}$. Since there are no defeasible rules, models and stable models coincide. Clearly any world satisfying both rules satisfies the propositional formula $x_1 \leftrightarrow x_2$, so the (stable) models of the theory base – there are two of them, $\{\emptyset\}$ and $\{\{x_1, x_2\}\}$ – correspond one-to-one to the models of the formula – $\emptyset$ and $\{x_1, x_2\}$. The second (stable) model, $\{\{x_1, x_2\}\}$, where we know that both atoms are true, might be undesired in a causal context such as that of Example 6.4. ◇

Here, our alternative intuition (DR) for strict rules comes into play. It is closer to the intuition behind defeasible rules and basically says that a strict rule is a directed inference rule on the knowledge level, and so we can use the same techniques for breaking strict cycles that we used earlier for defeasible ones. The formal definition simply says that with epistemic state Q, R (definitely possible worlds Q, potentially possible worlds R) in actual world w, a strict rule is satisfied if and only if knowing the truth of the premises implies the actual truth of its conclusion, where "knowing" refers to the conservative knowledge estimate given by the potentially possible worlds R:

$$Q, R, w \models r : \phi_1, \ldots, \phi_n \to \psi \text{ iff there is a } v \in R \text{ and } 1 \leq i \leq n \text{ with } v \not\models \phi_i \text{ or } w \models \psi$$

The remaining definitions, in particular those of models and stable models, stay the same.[11] This formal semantics of strict rules is just like that for defeasible rules, only without the additional condition that checks that the conclusion is not known to be false.

With this alternative semantics for strict rules, also positive strict cycles can be treated by the stable model semantics. However, there is another problem: this semantics is not able to produce the desired outcome of the "Married John" rule base.

Example 6.16 (Continued from Example 6.10). For the married John example and the intuition where strict rules are interpreted according to propositional material implication, we had two models, $Q_1 = \{\{g, w, b\}\}$ and $Q_2 = \{\{g, w, m, h\}\}$. Unfortunately, neither of the models persists when strict rules are interpreted according to our alternative intuition, where they are much closer to defeasible rules. Then, the semantics' eagerness to apply rules also applies to strict rules and leads to global inconsistency in the sense of allowing as the only model of the theory base the empty epistemic state. We exemplify this by showing that Q_1 is not a

[10] Roughly, for $i \geq 0$ we have $R_i \supseteq R_{i+1}$ whence for each stable model we have $Q = R_i$ for some $i \in \mathbb{N}$, furthermore it can be shown that $Q, Q, w \models r$ (Definition 6.6) iff $Q, w \models r$ (Definition 6.5).

[11] For the definition of a model the (DR) intuition uses the fact that $Q, w \models r$ iff $Q, Q, w \models r$.

model any more: Recall that Q_1 is a model iff $Q_1 = \{w \in W_A \mid Q_1, w \models TB\}$, in other words, if and only if $\{g, w, b\}$ is the one single world v for which we find $Q_1, v \models TB$. However, this is not the case. There is another world, $v' = \{g, w, b, m\}$, which satisfies the theory base in the epistemic state Q_1: First of all, the two strict rules r_1 and r_2 are satisfied by Q_1, v' since $v' \models w$ and $v' \models g$. We also have $Q_1, v' \models r_3 : b \to \neg h$ since $v' \models \neg h$. We find that $Q_1, v' \models r_4 : m \to h$ since $Q_1 \not\models m$. Finally, we can also show that $Q_1, v' \models r_5 : w \Rightarrow m$ because $v' \models m$; and that $Q_1, v' \models r_6 : g \Rightarrow b$ since $v' \models b$.

The problem is caused by r_4. Roughly speaking, there is incomplete knowledge about m – it is not known although it holds. In general, it is clear that the world v' should not be considered possible since in it John is a married bachelor. But the way strict rules are interpreted according to the alternative intuition, the semantics has no way to figure this out, because strict rules do not operate on the level of single worlds, but only through the interaction of epistemic states and single worlds. A similar thing happens for Q_2; likewise it can be verified that the theory has no models at all. ◇

This illustrates the difficulty of devising a semantics for defeasible theory bases that both possesses an eagerness to apply rules as well as it prevents self-supporting conclusions. Furthermore, our formalisation made it clear that the issue is linked to the question on which level strict rules should be enforced – on the level of single possible worlds or on theory level (knowledge level).

6.4.1 Relationship to Autoepistemic Logic

To explain the connection to related work in nonmonotonic reasoning, we briefly sketch how our possible-worlds semantics links to Moore's autoepistemic logic (AEL) (Moore, 1985). Propositional AEL enhances classical propositional logic by a unary modal connective **K** for knowledge. So for a formula φ, the AEL formula $\mathbf{K}\varphi$ stands for "φ is known." The semantics of autoepistemic logic is defined as follows: For a set B of formulas (the initial beliefs), a set T is an *expansion* of B if it coincides with the deductive closure of $B \cup \{\mathbf{K}\varphi \mid \varphi \in T\} \cup \{\neg\mathbf{K}\varphi \mid \varphi \notin T\}$. In words, T is an expansion if it equals what can be derived using the initial beliefs B and positive and negative introspection with respect to T itself.[12] The intuition behind **K** can be used to define a straightforward translation from theory bases into autoepistemic logic for the intuition (MI) behind strict rules.

Definition 6.7. Let $TB = (Lit, StrInf, DefInf)$ be a defeasible theory. Define an autoepistemic theory $\Omega(TB)$ as follows.

$$\begin{aligned} \Omega(TB) &= \{\Omega(r) \mid r \in StrInf \cup DefInf\} \\ \Omega(\phi_1, \ldots, \phi_n \to \psi) &= (\phi_1 \wedge \ldots \wedge \phi_n) \to \psi \\ \Omega(\phi_1, \ldots, \phi_n \Rightarrow \psi) &= (\mathbf{K}(\phi_1 \wedge \ldots \wedge \phi_n) \wedge \neg\mathbf{K}\neg\psi) \to \psi \end{aligned}$$ ◇

With this translation, theory base models according to Definition 6.5 correspond one-to-one to expansions of the resulting autoepistemic theory. Likewise, stable models of the theory base are in one-to-one correspondence with strong expansions of the autoepistemic theory, a constructive refinement of the original expansion semantics (Denecker et al., 2003).

For our alternative intuition (DR) for strict rules, the associated AEL translation is $\Omega(\phi_1, \ldots, \phi_n \to \psi) = (\mathbf{K}(\phi_1 \wedge \ldots \wedge \phi_n)) \to \psi$. It is readily seen that this translation is quite close to that of a defeasible rule. The relation of our intuition behind theory base semantics with default logic (Reiter, 1980) is immediate from reversing the translation of Konolige (1988).

[12] Moore himself also gave a possible-worlds based treatment of autoepistemic logic (Denecker et al., 2003).

6.4.2 Defining Further Semantics

The translation from defeasible theory bases into autoepistemic logic immediately provides us with the possibility to define further argumentation semantics in terms of possible-world structures. Up to now, we explicitly only considered possible-world structures $Q \subseteq W_A$ that were in a sense two-valued, that is, a possible world $w \in W_A$ was either considered an epistemic alternative ($w \in Q$) or not ($w \notin Q$). This is similar to the stable semantics in abstract argumentation, where each argument is either accepted or rejected. However, there are also three-valued abstract argumentation semantics, like the complete semantics, where the status of an argument might be neither accepted nor rejected, but undecided. To generalise such three-valued semantics to a possible-world setting, we need possible-worlds structures in which the epistemic status of a possible world w can be likewise undecided, that is, for all that is known, the world w *might* be an epistemic alternative.

Denecker et al. (2003) provided such a three-valued (even four-valued) possible-world treatment for autoepistemic logic. This treatment is embedded into the general algebraic framework of *approximation fixpoint theory* (Denecker et al., 2000). There, knowledge bases are associated with certain operators, and the semantics of the knowledge bases is then defined via fixpoints of these operators. In Chapter 3, we generalised several argumentation semantics to this abstract, operator-based setting. Applying these general definitions of semantics to the approximation operator for autoepistemic logic as defined by Denecker et al. (2003) immediately yields all of these semantics for defeasible theory bases. The pair $(2^{W_A}, \supseteq)$ is a complete lattice and its associated bilattice with information ordering is given by $(2^{W_A} \times 2^{W_A}, \sqsubseteq_i)$. This bilattice contains approximations of possible-world structures. The next definition is by Denecker et al. (2003) and specifies a model relation for approximative possible-world structures.

Definition 6.8. Let $P, S \subseteq W_A$, $w \in W_A$ and $a \in A$.

$(P,S), w \models a$	iff $a \in w$
$(P,S), w \models \varphi_1 \wedge \varphi_2$	iff both $(P,S), w \models \varphi_1$ and $(P,S), w \models \varphi_2$
$(P,S), w \models \varphi_1 \vee \varphi_2$	iff one of $(P,S), w \models \varphi_1$ or $(P,S), w \models \varphi_2$
$(P,S), w \models \neg\varphi$	iff $(S,P), w \not\models \varphi$
$(P,S), w \models \mathbf{K}\varphi$	iff for all $v \in P$, we have $(P,S), v \models \varphi$

For an autoepistemic theory Ω, we have $(P,S), w \models \Omega$ iff $(P,S), w \models \varphi$ for all $\varphi \in \Omega$. ◇

With this definition, it is immediate to define an approximation operator:

Definition 6.9. Let Ω be an autoepistemic theory over a propositional signature A. Consider $P, S \subseteq W_A$ and $w \in W_A$. Define the operator $\mathcal{D}_\Omega : 2^{W_A} \times 2^{W_A} \to 2^{W_A} \times 2^{W_A}$ by

$$\mathcal{D}_\Omega(P,S) = (\mathcal{D}'_\Omega(P,S), \mathcal{D}'_\Omega(S,P))$$
$$\mathcal{D}'_\Omega(P,S) = \{w \in W_A \mid (S,P), w \models \Omega\}$$ ◇

We can then use the definition of operator-based semantics from Chapter 3 (Table 3.1). We now give some examples to provide a glimpse of how some of the generalised semantics behave.

First of all, we want to note that most semantics for argumentation frameworks allow for more than one generalisation. We have seen this already in the case of the stable extension semantics, which can be generalised to ADFs in at least two ways, to models and *stable* models. Likewise, we presented two versions of two-valued epistemic semantics for defeasible theory bases. In the same vein, the grounded semantics for abstract argumentation can be generalised

in at least two ways: to the Kripke-Kleene semantics, the cycle-supporting version of the grounded semantics, and to the well-founded semantics, the cycle-rejecting version of the grounded semantics (Denecker et al., 2003).

Let us consider the rain/wet grass example (Example 6.5). There, the grounded (Kripke Kleene) semantics considers the world $\{rain, wet\}$ to be definitely possible, and all other worlds to be *potentially* possible. The grounded (well-founded) semantics for this example corresponds to the two-valued epistemic model given by W_A and considers all worlds to be definitely possible. Intuitively, the well-founded semantics does not derive any knowledge from the two mutually supporting defeasible rules (all possible worlds occur in all stable models), while the Kripke-Kleene semantics lends some more credence to the world where both rain and wet grass are true (because this one world occurs in both models, while all others only occur in one of them).

Example 6.17. Consider the literals $Lit = \{x_1, x_2, \neg x_1, \neg x_2\}$ and the theory base given by defeasible rules $DefInf = \{r_1 : \Rightarrow x_1, \quad r_2 : \Rightarrow \neg x_1\}$ and the strict rule $StrInf = \{r_3 : x_1 \rightarrow x_2\}$. It is clear that not both defeasible rules can be applied, so there are two different models: $Q_1 = \{\{x_1, x_2\}\}$ where r_1 has been applied, and r_3 then infers x_2; and $Q_2 = \{\emptyset, \{x_2\}\}$ where r_2 has been applied, and we do not know about x_2. In (both versions of) the grounded semantics of this theory base, no world is definitely considered possible, as the two models are disjoint. However, all the worlds in $Q_1 \cup Q_2$ are considered potentially possible. $\diamond$

Likewise, in Example 6.2, both versions of grounded semantics consider the three worlds $\{x_1, x_2, x_3\}$, $\{x_1, x_2, x_3, x_4\}$ and $\{x_1, x_2, x_3, x_5\}$ to be possible, albeit none of them definitely so. This shows that the generalised grounded semantics are not equal to sceptical reasoning among (stable) models, but rather an independent, weaker semantics. Indeed, this and other relationships between argumentation semantics carry over to their generalised versions.

6.5 Conclusion

In this chapter we presented a translation from theory bases to abstract dialectical frameworks. The translated frameworks satisfy the rationality postulates closure and direct/indirect consistency, which we generalised to make them independent of a specific target formalism. Furthermore, the translated frameworks can detect inconsistencies in the rule base and cyclic supports amongst literals. We also showed that the translation involves at most a quadratic blowup and is therefore effectively computable. In addition, our translation produces a number of statements which is linear in the size of the theory base and can be considered efficient in this regard. (In the approach of Caminada and Amgoud (2007) the number of produced arguments is unbounded in general.) In terms of desired behaviour, we compared our translation to previous approaches from the literature (Caminada and Amgoud, 2007; Wyner et al., 2009, 2013, 2015) and demonstrated how we avoid common problems. We also introduced possible-worlds semantics as a language to "think aloud" about defeasible theories. Along with this we presented two possible intuitions for strict rules and argued why we prefer one over the other. Of course, other intuitions are possible, and we mainly consider the present work a start for formulating intuitions in a formally precise way.

In earlier work, Brewka and Gordon (2010) translated Carneades (Gordon et al., 2007) argument evaluation structures (directly) to ADFs. They extended the original Carneades formalism by allowing cyclic dependencies among arguments. Meanwhile, Van Gijzel and Prakken (2011) also translated Carneades into AFs (via ASPIC+ (Prakken, 2010; Modgil and Prakken, 2013), that extends and generalises the definitions of Caminada and Amgoud (2007)).

They can deal with cycles, but there is only one unique grounded, preferred, complete, stable extension. Thus the semantic richness of abstract argumentation is not used, and the user cannot choose whether they want to accept or reject circular justifications of arguments. In contrast, in the approach of Brewka and Gordon (2010) the user can decide whether cyclic justifications should be allowed or disallowed, by choosing models or *stable* models as ADF semantics.

We regard the results of this chapter as another piece of evidence that abstract dialectical frameworks are well-suited as target formalisms for translations from rule-based nonmonotonic languages such as defeasible theory bases. A natural next step is to consider as input the specification language of ASPIC+ (Prakken, 2010; Modgil and Prakken, 2013), for which a recent approach to preferences between statements (Brewka et al., 2013) is a good starting point. In view of possible semantics for defeasible theories, it also seems fruitful to look at additional rationality postulates, for example those studied by Caminada et al. (2012) or Dung and Thang (2014). Further work could also encompass the study of further ADF semantics, like complete or preferred models (Brewka et al., 2013), and whether our translation to ADFs can be modified such that it is eager to apply defeasible rules and even coincides with our possible-world semantics.

Chapter 7

Discussion

In this thesis, we extensively, and in several regards also intensively, studied abstract dialectical frameworks (ADFs). Through defining their semantics via the notion of approximating operators, analysing their computational complexity and expressiveness, and applying ADFs to open issues in instantiated argumentation, we mapped out what ADFs can do, exemplified how they can be used, and at the same time explored their limits.

So what's next?

How can abstract argumentation be put to action?

At this point it might be useful to recall Walton's definition of an argument as (1) an object consisting of premises linked to a conclusion via an inference, that is (2) related to other arguments. While approaches to abstract argumentation are formidable in analysing the relationship between arguments, they decidedly ignore argument internals. And while this circumstance is often advertised as a strength of abstract argumentation, it is very well a significant limitation, as Walton (2009) so aptly recognises:

> It is important to emphasize that the use of such concepts and techniques [those of formal argumentation], while they have proved very valuable for teaching skills of critical thinking, have raised many problems about how to make the concepts and techniques more precise so that they can be applied more productively to realistic argumentation in natural language texts of discourse. Many of these problems arise from the fact that it can be quite difficult to interpret what is meant in a natural language text of discourse and precisely identify arguments contained in it. Ambiguity and vagueness are extremely common, and in many instances, the best one can do is to construct a hypothesis about how to interpret the argument based on the evidence given from the text of discourse.

Thus a lot of uncharted territory lies ahead of argumentation theory researchers.

7.1 Related and Possible Future Work

The reach of (some of) the formal results reported on herein has meanwhile exceeded the field of abstract argumentation and begun to influence neighbouring areas: In cooperation with Mario Alviano and Wolfgang Faber, we employed the computationally attractive properties of bipolar ADFs to aggregates in answer set programming. That has lead to a unifying view on polynomial-time decidable aggregates for the Pelov/Son-Pontelli stable model semantics for

logic programs with aggregates (Alviano, Faber, and Strass, 2016). It also shows that Brewka and Woltran's intuitions for defining bipolar ADFs were spot-on; at the same time it raises the question of how much the notion of bipolarity can be extended without sacrificing its nice computational properties. That is but one open question among many which are brought up by the work of others; in the remainder, we will therefore discuss related work on ADFs as well as emerging possibilities for future work.

Alviano and Faber (2015) compare several stable model semantics for ADFs and logic programs with aggregates. Their results show that when restricting to logic programs with exactly one rule for each atom, there is a perfect match of stable model semantics for the two formalisms: the PSP stable model semantics defined by Pelov (2004); Son and Pontelli (2007); Pelov, Denecker, and Bruynooghe (2007) coincides with the ultimate stable model semantics (Definition 3.7); the GZ stable model semantics defined by Gelfond and Zhang (2014) coincides with the approximate stable model semantics (Definition 3.2); the FLP stable model semantics (Faber, Pfeifer, and Leone, 2011) constitutes a new stable model semantics for ADFs, that Alviano and Faber (2015) call "F-stable" models.

Further Semantics Polberg et al. (2013) defined further extension-based semantics based on several novel, fine-grained notions of admissibility. Further novel extension-based semantics have been proposed by Polberg (2014). Polberg and Doder (2014) introduced probabilistic semantics for ADFs.

As hinted at in several places earlier, Gaggl and Strass (2014) generalised the (AF) *cf2* and *stage2* semantics to ADFs (based on ultimate asymmetric conflict-free interpretations). Further semantics that we already mentioned in some places are the grounded-fixpoint semantics (Bogaerts et al., 2015, Definition 6.8) and the F-stable models by Alviano and Faber (2015, Definition 10).

Other Work on ADFs Ellmauthaler and Wallner (2012) provided an implementation of ADFs which is based on answer set programming. Another such implementation was presented by Ellmauthaler and Strass (2014), with precursory work (Brewka et al., 2013; Ellmauthaler and Strass, 2013) and a recent continuation (Ellmauthaler and Strass, 2016). Diller et al. (2015) present an ADF implementation based on reasoning with quantified Boolean formulas.

In acclaimed work, Linsbichler (2014) applied the concept of *splitting* (Lifschitz and Turner, 1994; Turner, 1996; Baumann, 2011) to abstract dialectical frameworks. His ADF splitting method is identical to (if not slightly more general than) the ADF decomposition approach proposed at the same conference by Gaggl and Strass (2014).

Bochman (2016) explains the relationship of ADFs with the causal calculus of McCain and Turner (1997) and suggests possibilities for future extensions of ADFs. Polberg (2016) provides translations from other generalisations of Dung AFs into abstract dialectical frameworks, thereby giving further evidence on how ADFs can be successfully put to use in formal argumentation.

Bibliography

Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. PRIMES is in P. *Annals of Mathematics*, 160(2):781–793, 2004. doi: 10.4007/annals.2004.160.781.

Latifa Al-Abdulkarim, Katie Atkinson, and Trevor J. M. Bench-Capon. Abstract dialectical frameworks for legal reasoning. In Rinke Hoekstra, editor, *Proceedings of the Twenty-Seventh Annual Conference on Legal Knowledge and Information Systems (JURIX)*, volume 271 of *Frontiers in Artificial Intelligence and Applications*, pages 61–70. IOS Press, December 2014.

Latifah Al-Abdulkarim, Katie Atkinson, and Trevor J. M. Bench-Capon. Evaluating the use of abstract dialectical frameworks to represent case law. In *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Law (ICAIL)*, pages 156–160, June 2015.

Mario Alviano and Wolfgang Faber. Stable model semantics of abstract dialectical frameworks revisited: A logic programming perspective. In Qiang Yang and Michael Wooldridge, editors, *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2684–2690, Buenos Aires, Argentina, July 2015. IJCAI/AAAI.

Mario Alviano, Wolfgang Faber, and Hannes Strass. Boolean functions with ordered domains in answer set programming. In Dale Schuurmans and Michael Wellman, editors, *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI)*, pages 879–885, Phoenix, AZ, USA, February 2016.

Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.

Unnamed authors. Digital intuition: A computer program that can outplay humans in the abstract game of Go will redefine our relationship with machines. *Nature*, 529:437, 2016. doi: 10.1038/529437a. URL `http://www.nature.com/news/digital-intuition-1.19230`. Editorial.

Pietro Baroni and Massimiliano Giacomin. On principle-based evaluation of extension-based argumentation semantics. *Artificial Intelligence*, 171(10–15):675–700, 2007.

Ringo Baumann. Splitting an argumentation framework. In James P. Delgrande and Wolfgang Faber, editors, *Proceedings of the Eleventh International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR)*, volume 6645 of *LNCS*, pages 40–53. Springer-Verlag Berlin Heidelberg, 2011.

Ringo Baumann and Hannes Strass. On the number of bipolar Boolean functions. *Journal of Logic and Computation*, 2017. doi: 10.1093/logcom/exx025. Advance Access Online 07 August 2017.

Ringo Baumann, Wolfgang Dvořák, Thomas Linsbichler, Hannes Strass, and Stefan Woltran. Compact argumentation frameworks. In *Proceedings of the Twenty-First European Conference on Artificial Intelligence (ECAI)*, pages 69–74, Prague, Czech Republic, August 2014.

Ringo Baumann, Wolfgang Dvořák, Thomas Linsbichler, Christof Spanring, Hannes Strass, and Stefan Woltran. On rejected arguments and implicit conflicts: The hidden power of argumentation semantics. *Artificial Intelligence*, pages 244–284, 2016. doi: 10.1016/j.artint.2016.09.004. URL `http://www.sciencedirect.com/science/article/pii/S0004370216301102`.

Trevor J. M. Bench-Capon and Paul E. Dunne. Argumentation in Artificial Intelligence. *Artificial Intelligence*, 171(10–15):619–641, July 2007.

Philippe Besnard and Sylvie Doutre. Characterization of Semantics for Argument Systems. In Didier Dubois, Christopher A. Welty, and Mary-Anne Williams, editors, *Proceedings of the Ninth International Conference on Principles of Knowledge Representation and Reasoning (KR2004)*, pages 183–193. AAAI Press, 2004.

Nicole Bidoit and Christine Froidevaux. Negation by default and unstratifiable logic programs. *Theoretical Computer Science*, 78(1):85–112, 1991.

Alexander Bochman. Abstract dialectical argumentation among close relatives. In Pietro Baroni, Thomas F. Gordon, Tatjana Scheffler, and Manfred Stede, editors, *Computational Models of Argument – Proceedings of COMMA 2016, Potsdam, Germany, 12–16 September, 2016.*, volume 287 of *Frontiers in Artificial Intelligence and Applications*, pages 127–138. IOS Press, 2016. doi: 10.3233/978-1-61499-686-6-127. URL `http://dx.doi.org/10.3233/978-1-61499-686-6-127`.

Bart Bogaerts, Joost Vennekens, and Marc Denecker. Grounded fixpoints and their applications in knowledge representation. *Artificial Intelligence*, 224:51–71, 2015.

Richard Booth. Judgment aggregation in abstract dialectical frameworks. In Thomas Eiter, Hannes Strass, Mirosław Truszczyński, and Stefan Woltran, editors, *Advances in Knowledge Representation, Logic Programming, and Abstract Argumentation – Essays Dedicated to Gerhard Brewka on the Occasion of His 60th Birthday*, volume 9060 of *Lecture Notes in Computer Science*, pages 296–308. Springer-Verlag Berlin Heidelberg, 2015. doi: 10.1007/978-3-319-14726-0_20. URL `http://dx.doi.org/10.1007/978-3-319-14726-0_20`.

Ravi B. Boppana. Threshold functions and bounded depth monotone circuits. *Journal of Computer and System Sciences*, 32(2):222–229, 1986.

Nicolas Bourbaki. Sur le théorème de Zorn. *Archiv der Mathematik*, pages 434–437, 1949/50.

Michael Bowling, Neil Burch, Michael Johanson, and Oskari Tammelin. Heads-up limit hold'em poker is solved. *Science*, 347(6218):145–149, 2015. doi: 10.1126/science.1259433. URL `http://science.sciencemag.org/content/347/6218/145`.

Gerhard Brewka and Thomas F. Gordon. Carneades and abstract dialectical frameworks: A reconstruction. In *Proceedings of the Third International Conference on Computational Models of Argument (COMMA)*, volume 216 of *FAIA*, pages 3–12. IOS Press, September 2010.

Gerhard Brewka and Stefan Woltran. Abstract dialectical frameworks. In *Proceedings of the Twelfth International Conference on the Principles of Knowledge Representation and Reasoning (KR)*, pages 102–111, 2010.

Gerhard Brewka, Paul E. Dunne, and Stefan Woltran. Relating the semantics of abstract dialectical frameworks and standard AFs. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI)*, pages 780–785. IJCAI/AAAI, 2011.

Gerhard Brewka, Stefan Ellmauthaler, Hannes Strass, Johannes Peter Wallner, and Stefan Woltran. Abstract dialectical frameworks revisited. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence (IJCAI)*, pages 803–809. IJCAI/AAAI, August 2013.

Gerhard Brewka, Sylwia Polberg, and Stefan Woltran. Generalizations of Dung frameworks and their role in formal argumentation. *IEEE Intelligent Systems*, 29(1):30–38, 2014. ISSN 1541-1672. Special Issue on Representation and Reasoning.

Martin Caminada. On the Issue of Reinstatement in Argumentation. In *Proceedings of the Tenth European Conference on Logics in Artificial Intelligence*, volume 4160 of *Lecture Notes in Computer Science*, pages 111–123. Springer-Verlag Berlin Heidelberg, September 2006.

Martin Caminada. An Algorithm for Stage Semantics. In *Computational Models of Argument: Proceedings of COMMA 2010*, pages 147–158, 2010.

Martin Caminada and Leila Amgoud. On the evaluation of argumentation formalisms. *Artificial Intelligence*, 171(5–6):286–310, 2007.

Martin W.A. Caminada, Walter A. Carnielli, and Paul E. Dunne. Semi-stable Semantics. *Journal of Logic and Computation*, 22(5):1207–1254, 2012.

Claudette Cayrol and Marie-Christine Lagasquie-Schiex. Bipolar abstract argumentation systems. In Guillermo Simari and Iyad Rahwan, editors, *Argumentation in Artificial Intelligence*, pages 65–84. Springer-Verlag Berlin Heidelberg, 2009.

Keith L. Clark. Negation as failure. In Hervé Gallaire and Jack Minker, editors, *Logic and Data Bases*, pages 293–322. Plenum Press, 1978.

Sylvie Coste-Marquis, Caroline Devred, and Pierre Marquis. Symmetric argumentation frameworks. In Lluis Godo, editor, *Proceedings of the Eighth European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU)*, volume 3571 of *LNCS*, pages 317–328. Springer-Verlag Berlin Heidelberg, 2005.

Sylvie Coste-Marquis, Sébastien Konieczny, Jean-Guy Mailly, and Pierre Marquis. On the revision of argumentation systems: Minimal change of arguments statuses. In *Proceedings of the Fourteenth International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 52–61, 2014.

Adnan Darwiche and Pierre Marquis. A knowledge compilation map. *Journal of Artificial Intelligence Research*, 17:229–264, 2002.

B.A. Davey and H.A. Priestley. *Introduction to Lattices and Order*. Cambridge University Press, second edition, 2002.

Marc Denecker, D. Theseider-Dupré, and Kristof Van Belleghem. An Inductive Definition Approach to Ramifications. *Linköping Electronic Articles in Computer and Information Science*, 3(7):1–43, January 1998.

Marc Denecker, Victor Marek, and Mirosław Truszczyński. Approximations, Stable Operators, Well-Founded Fixpoints and Applications in Nonmonotonic Reasoning. In *Logic-Based Artificial Intelligence*, pages 127–144. Kluwer Academic Publishers, 2000.

Marc Denecker, V. Wiktor Marek, and Mirosław Truszczyński. Uniform Semantic Treatment of Default and Autoepistemic Logics. *Artificial Intelligence*, 143(1):79–122, 2003.

Marc Denecker, Victor W. Marek, and Mirosław Truszczyński. Ultimate approximation and its application in nonmonotonic knowledge representation systems. *Information and Computation*, 192(1):84–121, 2004.

Marc Denecker, Gerhard Brewka, and Hannes Strass. A formal theory of justifications. In Francesco Calimeri, Giovambattista Ianni, and Mirosław Truszczyński, editors, *Proceedings of the Thirteenth International Conference on Logic Programming and Non-monotonic Reasoning (LPNMR)*, pages 250–264, Lexington, KY, USA, September 2015. Springer-Verlag Berlin Heidelberg.

Martin Diller, Johannes Peter Wallner, and Stefan Woltran. Reasoning in abstract dialectical frameworks using quantified Boolean formulas. *Argument & Computation*, 6(2):149–177, 2015.

Yannis Dimopoulos and Alberto Torres. Graph Theoretical Structures in Logic Programs and Default Theories. *Theoretical Computer Science*, 170(1–2):209–244, 1996.

Yannis Dimopoulos, Bernhard Nebel, and Francesca Toni. On the computational complexity of assumption-based argumentation for default reasoning. *Artificial Intelligence*, 141(1/2):57–78, 2002.

Phan Minh Dung. On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and *n*-Person Games. *Artificial Intelligence*, 77:321–358, 1995.

Phan Minh Dung and Phan Minh Thang. Closure and consistency in logic-associated argumentation. *Journal of Artificial Intelligence Research*, 49:79–109, 2014.

Phan Minh Dung, Paolo Mancarella, and Francesca Toni. Computing ideal sceptical argumentation. *Artificial Intelligence*, 171(10):642–674, 2007.

Paul E. Dunne. The computational complexity of ideal semantics. *Artificial Intelligence*, 173(18): 1559–1591, 2009.

Paul E. Dunne and Trevor J. M. Bench-Capon. Coherence in Finite Argument Systems. *Artificial Intelligence*, 141(1/2):187–203, 2002.

Paul E. Dunne and Michael Wooldridge. Complexity of abstract argumentation. In Guillermo Simari and Iyad Rahwan, editors, *Argumentation in Artificial Intelligence*, pages 85–104. Springer-Verlag Berlin Heidelberg, 2009.

Paul E. Dunne, Wolfgang Dvořák, Thomas Linsbichler, and Stefan Woltran. Characteristics of multiple viewpoints in abstract argumentation. In *Proceedings of the Fourteenth International Conference on the Principles of Knowledge Representation and Reasoning (KR)*, pages 72–81, Vienna, Austria, July 2014.

Paul E. Dunne, Wolfgang Dvořák, Thomas Linsbichler, and Stefan Woltran. Characteristics of multiple viewpoints in abstract argumentation. *Artificial Intelligence*, 228:153–178, 2015.

Wolfgang Dvořák, Sebastian Ordyniak, and Stefan Szeider. Augmenting tractable fragments of abstract argumentation. *Artificial Intelligence*, 186:157–173, 2012.

Wolfgang Dvořák, Matti Järvisalo, Johannes P. Wallner, and Stefan Woltran. Complexity-Sensitive Decision Procedures for Abstract Argumentation. *Artificial Intelligence*, 206:53–78, 2014.

Sjur Kristoffer Dyrkolbotn. How to argue for anything: Enforcing arbitrary sets of labellings using AFs. In *Proceedings of the Fourteenth International Conference on the Principles of Knowledge Representation and Reasoning (KR)*, pages 626–629, Vienna, Austria, July 2014.

Uwe Egly, Sarah Alice Gaggl, and Stefan Woltran. Answer-set Programming Encodings for Argumentation Frameworks. *Argument and Computation*, 1(2):147–177, 2010.

Thomas Eiter and Georg Gottlob. On the computational cost of disjunctive logic programming: Propositional case. *Annals of Mathematics and Artificial Intelligence*, 15(3–4):289–323, 1995.

Thomas Eiter, Michael Fink, and João Moura. Paracoherent Answer Set Programming. In *Proceedings of the Twelfth International Conference on Principles of Knowledge Representation and Reasoning*, May 2010.

Thomas Eiter, Michael Fink, Jörg Pührer, Hans Tompits, and Stefan Woltran. Model-based recasting in answer-set programming. *Journal of Applied Non-Classical Logics*, 23(1–2):75–104, 2013.

Stefan Ellmauthaler and Hannes Strass. The DIAMOND system for argumentation: Preliminary report. In Michael Fink and Yuliya Lierler, editors, *Proceedings of the Sixth International Workshop on Answer Set Programming and Other Computing Paradigms (ASPOCP)*, September 2013.

Stefan Ellmauthaler and Hannes Strass. The DIAMOND System for Computing with Abstract Dialectical Frameworks. In Simon Parsons, Nir Oren, and Chris Reed, editors, *Proceedings of the Fifth International Conference on Computational Models of Argument (COMMA)*, volume 266 of *FAIA*, pages 233–240, The Scottish Highlands, Scotland, United Kingdom, September 2014. IOS Press.

Stefan Ellmauthaler and Hannes Strass. DIAMOND 3.0 – A native C++ implementation of DIAMOND. In Pietro Baroni, editor, *Proceedings of the Sixth International Conference on Computational Models of Argument (COMMA)*, volume 287 of *FAIA*, pages 471–472, Potsdam, Germany, September 2016. IOS Press.

Stefan Ellmauthaler and Johannes Peter Wallner. Evaluating Abstract Dialectical Frameworks with ASP. In *Computational Models of Argument: Proceedings of COMMA 2012*, pages 505–506, 2012.

Wolfgang Faber, Gerald Pfeifer, and Nicola Leone. Semantics and complexity of recursive aggregates in answer set programming. *Artificial Intelligence*, 175(1):278–298, 2011. doi: 10.1016/j.artint.2010.04.002. URL `http://dx.doi.org/10.1016/j.artint.2010.04.002`.

Melvin Fitting. Fixpoint Semantics for Logic Programming: A Survey. *Theoretical Computer Science*, 278(1–2):25–51, 2002.

Tim French, Wiebe van der Hoek, Petar Iliev, and Barteld Kooi. On the succinctness of some modal logics. *Artificial Intelligence*, 197:56–85, 2013.

Joel Friedman. Constructing $O(n \log n)$ size monotone formulae for the k-th elementary symmetric polynomial of n Boolean variables. *SIAM Journal on Computing*, 15:641–654, 1986.

Dov M. Gabbay. Dung's argumentation is essentially equivalent to classical propositional logic with the Peirce-Quine dagger. *Logica Universalis*, 5(2):255–318, 2011.

Dov M. Gabbay and Artur S. d'Avila Garcez. Logical Modes of Attack in Argumentation Networks. *Studia Logica*, 93(2–3):199–230, 2009.

Sarah A. Gaggl and Hannes Strass. Decomposing Abstract Dialectical Frameworks. In Simon Parsons, Nir Oren, and Chris Reed, editors, *Proceedings of the Fifth International Conference on Computational Models of Argument (COMMA)*, volume 266 of *FAIA*, pages 281–292. IOS Press, September 2014.

Sarah Alice Gaggl, Sebastian Rudolph, and Hannes Strass. On the computational complexity of naive-based semantics for abstract dialectical frameworks. In Qiang Yang and Michael Wooldridge, editors, *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2985–2991, Buenos Aires, Argentina, July 2015. IJCAI/AAAI.

M. Gebser, R. Kaminski, B. Kaufmann, M. Ostrowski, T. Schaub, and M. Schneider. Potassco: The Potsdam Answer Set Solving Collection. *AI Communications*, 24(2):105–124, 2011. Available at `https://potassco.org`.

Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In *Proceedings of the International Conference on Logic Programming (ICLP)*, pages 1070–1080. The MIT Press, 1988.

Michael Gelfond and Yuanlin Zhang. Vicious circle principle and logic programs with aggregates. *Theory and Practice of Logic Programming*, 14(4-5):587–601, 2014. doi: 10.1017/S1471068414000222. URL `http://dx.doi.org/10.1017/S1471068414000222`.

Goran Gogic, Henry Kautz, Christos Papadimitriou, and Bart Selman. The comparative linguistics of knowledge representation. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 862–869. Morgan Kaufmann, 1995.

Thomas F. Gordon, Henry Prakken, and Douglas Walton. The Carneades model of argument and burden of proof. *Artificial Intelligence*, 171(10–15):875–896, 2007.

Georg Gottlob. Translating Default Logic into Standard Autoepistemic Logic. *Journal of the ACM*, 42(4):711–740, 1995.

Davide Grossi. Fixpoints and Iterated Updates in Abstract Argumentation. In *Proceedings of the Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning (KR)*, pages 65–74. AAAI Press, 2012.

Hadassa Jakobovits and Dirk Vermeir. Robust Semantics for Argumentation Frameworks. *Journal of Logic and Computation*, 9(2):215–261, 1999.

Tomi Janhunen. On the Intertranslatability of Non-monotonic Logics. *Annals of Mathematics and Artificial Intelligence*, 27(1–4):79–128, 1999.

Stasys Jukna. *Boolean Function Complexity: Advances and Frontiers*, volume 27 of *Algorithms and Combinatorics*. Springer-Verlag Berlin Heidelberg, 2012.

Kurt Konolige. On the Relation Between Default and Autoepistemic Logic. *Artificial Intelligence*, 35(3):343–382, 1988.

Joohyung Lee. A model-theoretic counterpart of loop formulas. In Leslie Pack Kaelbling and Alessandro Saffiotti, editors, *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 503–508. Professional Book Center, aug 2005. URL `http://ijcai.org/Proceedings/05/Papers/1280.pdf`.

Vladimir Lifschitz and Alexander Razborov. Why are there so many loop formulas? *ACM Transactions on Computational Logic*, 7(2):261–268, April 2006.

Vladimir Lifschitz and Hudson Turner. Splitting a logic program. In Pascal Van Hentenryck, editor, *Logic Programming, Proceedings of the Eleventh International Conference on Logic Programming, Santa Marherita Ligure, Italy, June 13-18, 1994*, pages 23–37. MIT Press, 1994.

Vladimir Lifschitz, Lappoon R. Tang, and Hudson Turner. Nested expressions in logic programs. *Annals of Mathematics and Artificial Intelligence*, 25(3–4):369–389, 1999. ISSN 1012-2443. doi: 10.1023/A:1018978005636.

Fangzhen Lin and Yuting Zhao. ASSAT: Computing answer sets of a logic program by SAT solvers. *Artificial Intelligence*, 157(1-2):115–137, 2004.

Thomas Linsbichler. Splitting abstract dialectical frameworks. In Simon Parsons, Nir Oren, and Chris Reed, editors, *Proceedings of the Fifth International Conference on Computational Models of Argument (COMMA)*, volume 266 of *FAIA*, pages 357–368. IOS Press, September 2014.

V. Wiktor Marek and Mirosław Truszczyński. Autoepistemic logic. *Journal of the ACM*, 38(3): 587–618, 1991.

Norman McCain and Hudson Turner. Causal theories of action and change. In Benjamin Kuipers and Bonnie L. Webber, editors, *Proceedings of the Fourteenth National Conference on Artificial Intelligence and Ninth Innovative Applications of Artificial Intelligence Conference, AAAI 97, IAAI 97, July 27-31, 1997, Providence, Rhode Island.*, pages 460–465. AAAI Press / The MIT Press, 1997. URL `http://www.aaai.org/Library/AAAI/1997/aaai97-071.php`.

Sanjay Modgil. Reasoning about preferences in argumentation frameworks. *Artificial Intelligence*, 173(9–10):901–934, 2009.

Sanjay Modgil and Henry Prakken. A general account of argumentation and preferences. *Artificial Intelligence*, 195(0):361–397, 2013.

Robert Moore. Semantical Considerations of Nonmonotonic Logic. *Artificial Intelligence*, 25(1): 75–94, 1985.

Søren H. Nielsen and Simon Parsons. A generalization of Dung's abstract framework for argumentation: Arguing with sets of attacking arguments. In *Argumentation in Multi-Agent Systems*, volume 4766 of *LNCS*, pages 54–73. Springer, 2006.

Juan Carlos Nieves, Mauricio Osorio, and Claudia Zepeda. A Schema for Generating Relevant Logic Programming Semantics and its Applications in Argumentation Theory. *Fundamenta Informaticae*, 106(2–4):295–319, 2011.

Monica Nogueira, Marcello Balduccini, Michael Gelfond, Richard Watson, and Matthew Barry. An A-Prolog decision support system for the Space Shuttle. In Alessandro Provetti and Tran Cao Son, editors, *Answer Set Programming*, 2001.

Mauricio Osorio, Claudia Zepeda, Juan Carlos Nieves, and Ulises Cortés. Inferring acceptable arguments with answer set programming. In *Proceedings of the Sixth Mexican International Conference on Computer Science (ENC)*, pages 198–205, 2005.

Christos H. Papadimitriou. *Computational complexity*. John Wiley and Sons Ltd., 2003.

Nikolay Pelov. *Semantics of logic programs with aggregates*. PhD thesis, Katholieke Universiteit Leuven, Departement Computerwetenschappen, Celestijnenlaan 200A, 3001 Heverlee, Belgium, April 2004.

Nikolay Pelov, Marc Denecker, and Maurice Bruynooghe. Well-founded and stable semantics of logic programs with aggregates. *Theory and Practice of Logic Programming*, 7 (3):301–353, 2007. doi: 10.1017/S1471068406002973. URL `http://dx.doi.org/10.1017/S1471068406002973`.

Sylwia Polberg. Extension-based semantics of abstract dialectical frameworks. In Ulle Endriss and João Leite, editors, *Proceedings of the Seventh European Starting AI Researcher Symposium (STAIRS)*, volume 264 of *FAIA*, pages 240–249. IOS Press, August 2014.

Sylwia Polberg. Understanding the abstract dialectical framework. In Loizos Michael and Antonis C. Kakas, editors, *Proceedings of the Fifteenth European Conference on Logics in Artificial Intelligence (JELIA)*, volume 10021 of *Lecture Notes in Computer Science*, pages 430–446. Springer-Verlag Berlin Heidelberg, November 2016. doi: 10.1007/978-3-319-48758-8_28. URL `http://dx.doi.org/10.1007/978-3-319-48758-8_28`.

Sylwia Polberg and Dragan Doder. Probabilistic abstract dialectical frameworks. In Eduardo Fermé and João Leite, editors, *Proceedings of the Fourteenth European Conference on Logics in Artificial Intelligence (JELIA)*, volume 8761 of *Lecture Notes in Computer Science*, pages 591–599. Springer-Verlag Berlin Heidelberg, September 2014.

Sylwia Polberg, Johannes P. Wallner, and Stefan Woltran. Admissibility in the abstract dialectical framework. In João Leite, Tran Cao Son, Paolo Torroni, Leon van der Torre, and Stefan Woltran, editors, *Proceedings of the Fourteenth International Workshop on Computational Logic in Multi-Agent Systems (CLIMA XIV)*, volume 8143 of *LNAI*, pages 102–118. Springer-Verlag Berlin Heidelberg, 2013.

John L Pollock. Defeasible reasoning. *Cognitive Science*, 11(4):481–518, 1987.

Henry Prakken. An abstract framework for argumentation with structured arguments. *Argument & Computation*, 1(2):93–124, 2010.

Henry Prakken and Giovanni Sartor. A System for Defeasible Argumentation, with Defeasible Priorities. In *Artificial Intelligence Today*, LNAI, pages 365–379. Springer-Verlag Berlin Heidelberg, 1999.

Jörg Pührer. Realizability of three-valued semantics for abstract dialectical frameworks. In Qiang Yang and Michael Wooldridge, editors, *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 3171–3177. IJCAI/AAAI, Buenos Aires, Argentina, July 2015.

Raymond Reiter. A Logic for Default Reasoning. *Artificial Intelligence*, 13:81–132, 1980.

Domenico Saccà and Carlo Zaniolo. Deterministic and Non-Deterministic Stable Models. *Journal of Logic and Computation*, 7(5):555–579, 1997.

Jonathan Schaeffer, Neil Burch, Yngvi Björnsson, Akihiro Kishimoto, Martin Müller, Robert Lake, Paul Lu, and Steve Sutphen. Checkers is solved. *Science*, 317(5844):1518–1522, 2007. doi: 10.1126/science.1144079. URL `http://science.sciencemag.org/content/317/5844/1518`.

Yuping Shen and Xishun Zhao. Canonical logic programs are succinctly incomparable with propositional formulas. In *Proceedings of the Fourteenth International Conference on the Principles of Knowledge Representation and Reasoning (KR)*, pages 665–668, Vienna, Austria, July 2014.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529:484–489, 2016. doi: 10.1038/nature16961. URL `http://dx.doi.org/10.1038/nature16961`.

Tran Cao Son and Enrico Pontelli. A constructive semantic characterization of aggregates in answer set programming. *Theory and Practice of Logic Programming*, 7(3):355–375, 2007.

Hannes Strass. Approximating operators and semantics for abstract dialectical frameworks. *Artificial Intelligence*, 205:39–70, December 2013.

Hannes Strass. Expressiveness of two-valued semantics for abstract dialectical frameworks. *Journal of Artificial Intelligence Research*, 54:193–231, 2015a.

Hannes Strass. The relative expressiveness of abstract argumentation and logic programming. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI)*, pages 1625–1631, Austin, TX, USA, January 2015b.

Hannes Strass. Instantiating rule-based defeasible theories in abstract dialectical frameworks and beyond. *Journal of Logic and Computation*, February 2015c. Advance Access published 11 February 2015, `http://dx.doi.org/10.1093/logcom/exv004`.

Hannes Strass and Johannes Peter Wallner. Analyzing the computational complexity of abstract dialectical frameworks via approximation fixpoint theory. *Artificial Intelligence*, 226: 34–74, 2015.

Alfred Tarski. A Lattice-Theoretical Fixpoint Theorem and Its Applications. *Pacific Journal of Mathematics*, 5(2):285–309, 1955.

Francesca Toni and Marek Sergot. Argumentation and answer set programming. In M. Balduccini and T. Son, editors, *Logic Programming, Knowledge Representation, and Nonmonotonic Reasoning: Essays in Honor of Michael Gelfond*, volume 6565 of *LNAI*, pages 164–180. Springer, 2011.

G. S. Tseitin. On the complexity of derivations in the propositional calculus. *Structures in Constructive Mathematics and Mathematical Logic, Part II, Seminars in Mathematics (translated from Russian)*, pages 115–125, 1968.

Hudson Turner. Splitting a default theory. In William J. Clancey and Daniel S. Weld, editors, *Proceedings of the Thirteenth National Conference on Artificial Intelligence and Eighth Innovative Applications of Artificial Intelligence Conference, AAAI 96, IAAI 96, Portland, Oregon, August 4-8, 1996, Volume 1.*, pages 645–651. AAAI Press / The MIT Press, 1996. URL `http://www.aaai.org/Library/AAAI/1996/aaai96-096.php`.

Bas Van Gijzel and Henry Prakken. Relating Carneades with abstract argumentation. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence – Volume Two*, pages 1113–1119. IJCAI/AAAI, 2011.

Joost Vennekens, David Gilis, and Marc Denecker. Splitting an operator: Algebraic modularity results for logics with fixpoint semantics. *ACM Transactions on Computational Logic*, 7(4): 765–797, 2006.

Bart Verheij. Two approaches to dialectical argumentation: admissible sets and argumentation stages. In J.-J. Ch. Meyer and L.C. van der Gaag, editors, *Proceedings of the Eighth Dutch Conference on Artificial Intelligence (NAIC'96)*, pages 357–368, 1996.

Toshiko Wakaki and Katsumi Nitta. Computing Argumentation Semantics in Answer Set Programming. In *Proceedings of the Annual Conference of Japanese Society for Artificial Intelligence (JSAI)*, pages 254–269, 2008.

Johannes P. Wallner. *Complexity Results and Algorithms for Argumentation – Dung's Frameworks and Beyond*. PhD thesis, Vienna University of Technology, Institute of Information Systems, 2014. URL `http://permalink.obvsg.at/AC11706119`.

Douglas Walton. Argumentation theory: A very short introduction. In *Argumentation in Artificial Intelligence*. Springer Dordrecht Heidelberg London New York, 2009.

Yining Wu, Martin Caminada, and Dov M. Gabbay. Complete Extensions in Argumentation Coincide with 3-Valued Stable Models in Logic Programming. *Studia Logica*, 93(2–3):383–403, 2009.

Adam Wyner, Trevor Bench-Capon, and Paul Dunne. Instantiating knowledge bases in abstract argumentation frameworks. In *Proceedings of the AAAI Fall Symposium – The Uses of Computational Argumentation*, 2009.

Adam Wyner, Trevor J. M. Bench-Capon, and Paul E. Dunne. On the instantiation of knowledge bases in abstract argumentation frameworks. In *Proceedings of CLIMA XIV*, volume 8143 of *LNAI*, pages 34–50. Springer-Verlag Berlin Heidelberg, September 2013.

Adam Wyner, Trevor Bench-Capon, Paul Dunne, and Federico Cerutti. Senses of 'argument' in instantiated argumentation frameworks. *Argument & Computation*, 6(1):50–72, 2015.

Index

S defends a, 11
a-conjugate of X, 108

abstract dialectical framework (ADF), 12
ADF $D(F)$ associated to an AF F, 14
admissible pair for an operator, 46
admissible pair in an ADF, 29
admissible set for an AF, 11
AF associated to φ, 121
antimonotone operator, 7
approximate operator, 22
approximate stable model of an ADF, 27
approximating operator, 8
approximating operators, 8
argumentation stage, 49
atoms of a logic program, 6
attacking, 103
attacking link in an ADF, 13

bilattice, 8
bipolar ADF, 13
BW-admissible set of an ADF, 14
BW-preferred model of an ADF, 14
BW-stable model of a bipolar ADF, 13
BW-well-founded model of an ADF, 14

canonical approximating operator $\mathcal{O}$, 8
canonical logic program rule, 6
characteristic function of an AF, 11
closed under a set of strict rules in a DTB, 129
closure of M under $StrInf$, 129
closure postulate, 130
coherent pair, 43
compact translation, 98
complete extension of an AF, 11
complete lattice, 7
complete stable operator for $\mathcal{O}$, 9
conclusion of a rule in a DTB, 128
conflict-free pair for an operator (asymmetric), 47

conflict-free pair for an operator (symmetric), 48
conflict-free set of an ADF, 13
conflict-free set of arguments, 11
consistent pair, 8
consistent set of literals, 128

defeasible rule of a DTB, 128
defeasible theory, 129
defeasible theory base, 129
definite logic program rule, 6
dependent, 112
difference, 105
direct consistency postulate, 131
doubly-negated logic program literals, 6

epistemic state, 140
exact pair, 8
Example 6.2, 132
    Example 6.3, 134
    Example 6.11, 142
Example 2.2, 12
    Example 2.3, 13
    Example 2.4, 14
    Example 3.7, 30
    Example 3.8, 31
Example 3.15, 47
    Example 3.16, 47
Example 2.5, 15
Example 6.7, 137
    Example 6.12, 142
Example 6.1, 129
    Example 6.10, 141
    Example 6.16, 144
Example 6.5, 136
    Example 6.13, 142
    Example 6.14, 143
Example 6.6, 136
    Example 6.15, 143
Example 6.8, 141
    Example 6.9, 141
expansion, 145

faithful translation, 98
finitary AF, 11
fixpoint, 7

grounded extension of an AF, 11

indirect consistency postulate, 131
information ordering $\leq_i$, 8

JV-labelling, 40

L-conflict-free pair for an operator, 50
L-stable model of a logic program, 10
L-stable pair for an operator, 46
L-supported model of a logic program, 10
L-supported pair for an operator, 46
links of an ADF, 12
logic program (LP), 6

M-conflict-free pair for an operator, 49
M-stable model of a logic program, 10
M-supported model of a logic program, 10
model for *TB*, 141
model for an ADF, 13
monotone operator, 7

naive extension of an AF, 11
negation-as-failure literals over a logic program signature, 6
negative, 104
negative body atoms of a normal logic program rule, 6
normal logic program rule, 6

partial valuation of φ by (X, Y), 58
partially ordered set (poset), 7
polarity, 104
positive, 104
positive body atoms of a normal logic program rule, 6
postfixpoint, 7
preferred extension of an AF, 11
prefixpoint, 7

range of an AF extension, 11
realisation, 98
reliable pair for an operator $\mathcal{O}$, 29
roof, 51
rule body in a DTB, 128
rule head in a DTB, 128
rule name in a DTB, 128

semantically bipolar, 103
semi-stable extension of an AF, 11
set of worlds over A, 140
signature of a logic program, 6
stable extension of an AF, 11
stable model, 35
stable model for *TB*, 143
stable operator $\mathcal{SO}$, 8
stable operator for $\mathcal{O}$, 9
stage extension of an AF, 11

standard logic program $P(D)$ of an ADF D, 31
statements of an ADF, 12
strict rule of a DTB, 128
supported model of a logic program, 10
supporting, 103
supporting link in an ADF, 13
symmetric operator, 8
syntactically bipolar, 104

theory base, 129
three-valued interpretations, 35
three-valued stable model of a logic program, 10
total JV-labelling, 40
truth ordering $\leq_t$, 8
two-valued interpretation, 6
two-valued model for an ADF, 13

ultimate admissible, 34
ultimate approximation, 9
ultimate complete, 34
ultimate family of ADF semantics, 34
ultimate grounded, 34
ultimate preferred, 34

vocabulary, 101

well-formed extension of a theory-based AF (Wyner et al., 2013), 132
well-founded model of a logic program, 10
world, 140