FAKULTÄT FÜR !NFORMATIK

Faculty of Informatics

Solving Reasoning Problems on Abstract Dialectical Frameworks via Quantified Boolean Formulas

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Master of Science (M.Sc.)

im Rahmen des Studiums

Computational Logic

eingereicht von

Martin Diller
Matrikelnummer 1228429

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung: Assoc. Prof. Dipl.-Ing. Dr.techn. Stefan Woltran
Mitwirkung: Dipl.-Ing. Johannes Peter Wallner

Wien, 13.03.2014

(Unterschrift Verfasser)

(Unterschrift Betreuung)

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.ac.at

Solving Reasoning Problems on Abstract Dialectical Frameworks via Quantified Boolean Formulas

MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Master of Science (M.Sc.)

in

Computational Logic

by

Martin Diller
Registration Number 1228429

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: Assoc. Prof. Dipl.-Ing. Dr.techn. Stefan Woltran
Assistance: Dipl.-Ing. Johannes Peter Wallner

Vienna, 13.03.2014

(Signature of Author) (Signature of Advisor)

Erklärung zur Verfassung der Arbeit

Martin Diller
Donaufelder Str. 54, 1210 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit - einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

(Ort, Datum)	(Unterschrift Verfasser)

Acknowledgements

First of all, I want to thank my advisor, Stefan Woltran, for his extremely good disposition throughout the work on the thesis as well as his insights and guidance which have been of great value to me. I also want to thank my co-advisor, Johannes P. Wallner, for his kind and meticulous hands-on assistance in every aspect of the thesis, from formal to technical.

I also want to acknowledge here the help of Florian Lonsing with many issues having to do with `DepQBF` and QSAT, as well as Stefan Ellmauthaler for his support with `DIAMOND` and sharing his work with me.

I am grateful to the authorities and team of the European Master of Computational Logic as well as the Erasmus Mundus Programme for making my studies possible. I am also in profound debt with all of you back home who have supported and made possible my abrupt switch to studies in Europe, despite this meaning that I broke several commitments I had in Córdoba.

A very special thank you goes to my mother, Ana, and father, Guillermo, for their unconditional support, my grandmother, Marcela, and sister, Carolina, for their care, and beloved Marina for keeping very close. Finally, I also want to express my gratitude to my friends in Vienna and from the EMCL.

Abstract

"Abstract argumentation" is a subfield of the interdisciplinary area of study "argumentation theory" that has developed into an increasingly important area of computer science and artificial intelligence in the last two decades. Starting with the "argumentation frameworks" (AFs) proposed by P. M. Dung in 1995, several different formalisms for abstract argumentation have been devised to date with just as many desiderata in mind, the overarching focus of study of this field though still being to determine when an argument (or set of arguments) can be considered to remain undefeated or even come out victorious in the context of a dispute.

All formalisms for abstract argumentation include a means of representing arguments and their relationships ("a framework"), several methods to determine which arguments can be accepted together based on the structure of the frameworks (the "semantics"), as well as "reasoning tasks" that are defined in terms of these frameworks and semantics. One of the most general formalisms is that of "abstract dialectical frameworks" (ADFs), which is of a relatively recent date and allows for the direct representation of complex relations of support and attack between arguments through boolean formulas ("acceptance conditions") associated to the arguments. This increase in expressive power with respect to AFs has the consequence that many of the reasoning tasks that can be defined for ADFs suffer from an even "higher complexity" than the tasks they generalise in the context of AFs, being complete for up to the third level of the polynomial hierarchy.

Quantified boolean logic on the other hand is a relatively well established formalism in computer science, being of special relevance because of its connection with the polynomial hierarchy of complexity classes. Given the advances in the performance of software systems for this logic in recent years, it is also increasingly recognized to be a promising formalism in which to translate computational problems located within the polynomial hierarchy.

In this work we present complexity-sensitive reductions or encodings of some of the main reasoning tasks defined for ADFs with respect to some of the major semantics into the satisfiability problem of quantified boolean logic (QSAT). In this manner we provide a uniform axiomatization of these reasoning tasks into quantified boolean logic. Moreover, we pave the way for implementations of these reasoning tasks via the already mentioned advances in technologies for QSAT. Finally, we also describe a prototype of such a system we have implemented and report on preliminary experiments that serve as a first benchmark for implementations of these tasks via QSAT.

Kurzfassung

“Abstrakte Argumentation”, als Teilgebiet des interdisziplinären Forschungsfeld der “Argumentationstheorie”, hat während der letzten beiden Dekaden innerhalb der Computerwissenschaften und der Künstlichen Intelligenz immer mehr an Bedeutung gewonnen. Ausgehend vom Konzept der “Argumentation Frameworks” (AFs), entwickelt von P. M. Dung im Jahre 1995, wurden viele verschiedene Formalismen für abstrakte Argumentation entworfen. In all diesen Formalismen gilt es zu entscheiden unter welchen Umständen ein Argument in einer Debatte “akzeptabel” ist bzw. wann eine Menge von Argumenten in sich “akzeptabel” ist. Alle Formalismen für abstrakte Argumentation beinhalten Konzepte für die Repräsentation von Argumenten, deren Relation untereinander (zusammengenommen also das “Framework”), verschiedene Methoden um anhand der Struktur des Frameworks festzustellen, welche Argumente zu akzeptieren sind (die “Semantiken”), sowie verschiedene Möglichkeiten von Schlüssen, welche mittels der Frameworks und deren Semantiken gezogen werden können.

Einer der allgemeinsten Formalismen in diesem Kontext sind die kürzlich entwickelten “Abstract Dialectical Frameworks” (ADFs), welche eine direkte Repräsentation von komplexen Relationen zwischen den Argumenten mittels boolescher Formeln (“acceptance conditions”) ermöglichen. Diese – im Vergleich zu AFs – erhöhte Ausdruckstärke bedingt, dass verschiedene Schlussweisen auf ADFs nun eine höhere Komplexität aufweisen, als jene Schlussweisen auf AFs, welche sie verallgemeinern. Manche Schlussweisen sind sogar vollständig für die dritte Stufe der polynomiellen Hierarchie.

Quantifizierte boolesche Logik auf der anderen Seite ist ein etablierter Formalismus in den Computerwissenschaften, welche eine direkte Verbindung mit der polynomiellen Hierarchie der Komplexitätsklassen hat. Durch den Fortschritt der Software Systeme für diese Logik hat sich die quantifizierte boolesche Logik als vielversprechender Formalismus, um Probleme innerhalb der polynomiellen Hierarchie zu lösen, herauskristallisiert.

In dieser Arbeit präsentieren wir Reduktionen, oder Kodierungen, von einigen der Schlussweisen in ADFs für die wichtigsten Semantiken auf das Problem der Erfüllbarkeit von quantifizierten booleschen Formeln (QSAT), welche die Komplexität der ursprünglichen Probleme berücksichtigen. Auf diese Art stellen wir eine uniforme Axiomatisierung für diese Schlussweisen in quantifizierter boolescher Logik bereit. Weiters bahnen wir den Weg für Implementierungen dieser Schlussweisen, da sie die fortschrittlichen Techniken, welche für QSAT entwickelt wurden, nutzen können. Schlussendlich beschreiben wir einen von uns implementierten Prototypen eines solchen Systems und diskutieren Ergebnisse erster Experimente.

Contents

CHAPTER 1

Introduction

Broady construed, the "study of argumentation may (...) be considered as concerned with how assertions are proposed, discussed, and resolved in the context of issues upon which several diverging opinions may be held" [Bench-Capon and Dunne, 2007]. From a normative perspective which is especially relevant for the development of computational applications, the interest can also be on how such assertions *should* be proposed, discussed, resolved, etc. in view of some objective such as obtaining information, persuasion or reaching a decision. Characterized in this general manner, the study of argumentation has a history as long as that of logic, argumentation theory itself today being viewed as an interdisciplinary field at the intersection between several disciplines (including, for example, legal theory, economics, linguistics, cognitive science, philosophy, and artificial intelligence). Nevertheless, studies initiated around the 1950's within philosophy have played a particulary significant role in the establishment of this research area [Toulmin, 2003, Johnson and Blair, 1994, Pollock, 1987].

In the case of artificial intelligence and computer science, the development into a relatively distinct and increasing important field has only ocurred in the last two decades. [Chesñevar et al., 2000] and [Bench-Capon and Dunne, 2007] review the history of this development, giving an overview of the main research directions as well as the connections between the study of argumentation from a computational perspective and development in other fields, especially philosophy. Today several research directions exist, in some cases attending to different phases of the "argumentation process" as described at the beginning of the previous paragraph as well as different types of arguments and argumentation contexts. Relatively recent books attempting to give an overview of this field are, for example, [Besnard and Hunter, 2008] and [Rahwan and Simari, 2009].

Argumentation has also increasingly received attention in artificial intelligence and computer science because of the connections between argumentation and other well established areas of computer science and artificial intelligence such as knowledge representation [Prakken and Sartor, 1997, Amgoud and Cayrol, 2002, Besnard and Hunter, 2005] and the foundations of multi agent systems [Amgoud et al., 2005, Kakas and Moraitis, 2006, McBurney et al., 2012]. Finally, there are several application domains such as in multi agent systems, decision support systems [Amgoud and Prade, 2009], computational assistance to legal reasoning [Bench-Capon et al., 2009], electronic governance [Cartwright and Atkinson, 2009], electronic health tools [Tullio and Grasso, 2011], etc.

The work by Dung [Dung, 1995] on abstract argumentation, in particular, is usually seen as a significant landmark in the consolidation of the field of argumentation in computer science and artificial intelligence [Bench-Capon and Dunne, 2007] and this area has, arguably, recieved the most attention from researchers in these disciplines to the present date. From the perspective of argumentation, the central concern of abstract argumentation is the evaluation of a set of arguments and their relations, what is called an "argumentation framework", to be able to extract subsets of the arguments that can all be accepted together from some point of view. Particular arguments can then be deemed to be acceptable if, for example, they form part of one of such "coherent viewpoints". Especially interesting for the area of knowledge representation is that the entailment relation of various important formalisms for non-monotonic reasoning have been translated into the abstract argumentation framework [Bondarenko et al., 1993, Dung, 1995, Bondarenko et al., 1997, Chesñevar et al., 2000, Prakken and Vreeswijk, 2002, Strass, 2013], a fact that is considered in [Dung, 1995] to be evidence for the "correctness" or "appropriateness" of the proposed model of argumentation.

What makes the approach of abstract argumentation *abstract* is that acceptance of arguments is decided based only on the topology of the argumentation framework, i.e. the internal structure of the arguments is disregarded. The criteria or method used to settle the acceptance of arguments is called a "semantics". By now a plethora of semantics motivated by different theoretical and practical concerns have been developed for this framework [Baroni et al., 2011a] and one important direction of research has been to determine properties of these semantics that also enable providing principles that allow for comparison between them [Baroni and Giacomin, 2007].

In the work of [Dung, 1995], argumentation frameworks are directed graphs where nodes represent arguments and links correspond to one argument attacking another. Although very general, this model does not directly support modelling more complex interactions between arguments as relations of mutual support and more sophisticated relations of attack. For this and similar issues, several extensions of the AF framework have been proposed to date (e.g. those presented in [Bench-Capon, 2003, Cayrol and Lagasquie-Schiex, 2005, Coste-Marquis et al., 2006, Baroni et al., 2011b]), one of the most general being that of abstract dialectical frameworks (ADFs) [Brewka and Woltran, 2010].

In this last framework, acceptance conditions in the form of arbitrary boolean formulas are associated to every argument. In [Brewka and Woltran, 2010] the most important semantics defined for argumentation frameworks as defined by Dung are generalised to ADFs, although the so called "preferred" and "stable" semantics are restricted to a type of ADFs called bipolar. In [Brewka et al., 2013] the semantics are extended to cover all ADFs.

Various reasoning tasks can be defined on abstract argumentation frameworks, some of the central ones being those that evaluate the "acceptability" of an argument with respect to a given framework and semantics. Already in the scenario of Dung's argumentation frameworks, most of the reasoning tasks have been shown to suffer from high computational complexity (e.g. see [Dunne and Bench-Capon, 2002]), although most remain within what in complexity theory has been identified as the "second level of the polynomial hierarchy" [Stockmeyer, 1976]. For ADFs, the complexity of many of the main reasoning tasks with respect to the generalised semantics jump one level of the polynomial hierarchy [Strass and Wallner, 2013, Strass and Wallner, 2014], resulting in some of the reasoning tasks even being on the third level of the polynomial hierarchy.

Given the situation described in the previous paragraph, the question of how to implement reasoning with respect to abstract argumenation frameworks becomes significant from a computational perspective and has received considerable attention for Dung's frameworks. A good survey of the approaches and advances in this direction is [Charwat et al., 2013] which classifies the two main approches into "reduction based" and "direct" approaches. Direct approaches are tailored specifically to reasoning on abstract argumentation systems, some of the main instances of this approach being so called "labelling-based algorithms" [Doutre and Mengin, 2001, Modgil and Caminada, 2009, Nofal et al., 2012, Verheij, 2007], characterisations via "dialogue-games" [Modgil and Caminada, 2009, Thang et al., 2009] or "dynamic programming" based methods [Dvořák et al., 2012a]. "Reduction based approaches" on the other hand aim to translate the reasoning problems of the field of argumentation to some other formalism for which efficient systems exist, for example propositional logic [Besnard and Doutre, 2004, Egly and Woltran, 2006, Arieli and Caminada, 2013, Dvořák et al., 2012], answer set programming (see [Toni and Sergot, 2011] for a survey on this approach) or constraint satisfaction problems [Amgoud and Devred, 2011, Bistarelli and Santini, 2011].

The two main software systems existing to date for reasoning on ADFs are instances of the "reduction based" approach and use answer set programming (ASP) as the target formalism, the first one, `ADFSys` [Ellmauthaler and Wallner, 2012, Ellmauthaler, 2012] having the semantics of [Brewka and Woltran, 2010] as a basis, while the second one, `DIAMOND` [Ellmauthaler and Strass, 2013], implements evaluation of ADFs with respect to the revised and generalised semantics given in [Brewka et al., 2013]. Although using answer set programming as a target formalism for implementing abstract argumentation can lead to relatively simple programs and enables harnessing the power of efficient answer set solvers, reasoning tasks beyond the first level of the polynomial hierarchy can only be implemented by making use of different fragments of ASP which allow for higher expressivity but the combination of which can lead to some technical difficulties. `DIAMOND` avoids some of these difficulties by assuming a representation of ADFs where acceptance conditions are represented as boolean functions instead of boolean formulas, but as a consequence the reductions to ASP defined by this system are, strictly speaking, not efficient since the transformation of boolean formulas to the boolean functions they represent is exponential in general.

In this work we present translations or reductions of some of the main reasoning tasks defined for ADFs into the problem of the satisfiability of quantified boolean formulas (QBFs) or QSAT. In this manner we provide a *uniform* axiomatization of these reasoning tasks into the relatively well established formalism of quantified boolean logic. Moreover, for the decision problems we consider, the reductions we present are complexity-sensitive in the sense that the reductions are efficient (polynomial) and that the complexity of QSAT of the target class of formulas is not "harder" than that of the reasoning task being encoded. In this manner, we also provide a theoretical foundation for the implementation of reasoning systems for ADFs via the existence of increasingly efficient solvers that have been developed for QSAT (see, for example, the results presented in [Peschiera et al., 2010, Lonsing and Seidl, 2013]). Finally, we motivate and present a prototype of such a reasoning system we have implemented using the QSAT solver `DepQBF` [Lonsing and Biere, 2010a] as back-end. We also report on preliminary experiments on this prototype system which serves as an "existence-proof" of the feasibility of this approach, although further empirical evaluation is necessary to determine more clearly its potential with respect to other approaches for implementing reasoning on ADFs.

Quantified boolean logic is an extension of propositional logic, allowing for quantification over propositional atoms. Research in this fomalism in computer science and in particular, in complexity theory, gained special momentum once Meyer and Stockmeyer [Stockmeyer and Meyer, 1973] showed that QSAT is complete for the complexity class PSPACE, which consists of all decision problems that can be solved by deterministic Turing machines in polynomial space in the size of the problem instance. Another important finding has been that of the existence of an important link between QSAT restricted to types of QBFs classified according to the structure of quantification and the hierarchy of complexity classes contained in PSPACE introduced by the same authors (see also [Stockmeyer, 1976]), the so called "polynomial hierarchy" that we have already referred to above. Specifically it was shown also in this work that each of these types of QBFs is complete for a corresponding class within the polynomial hierarchy. These results not only provide a foundation for complexity *analysis* of computational problems, by allowing to determine lower bounds for the hardness of a problem by providing reductions of types of QBFs to the problems in question, but also imply that for problems that are known to be in PSPACE an efficient reduction to some of the mentioned types of QBFs must exist.

In recent years ever more efficient solvers for QBFs have been presented (apart from `DepQBF`, see, for example, those presented in [Giunchiglia et al., 2010a, Klieber et al., 2010, Janota et al., 2012, Goultiaeva and Bacchus, 2013]), making reductions of problems to QBFs not only theoretically interesting but also a viable strategy for implementations of solvers for these problems. Research in QSAT solving strategies has been very influenced by strategies for solving the analogous problem for propositional formulas, SAT. It started with work by Kleine, Buening, et al. [Büning et al., 1995] on the generalisation of the resolution principle for SAT to the QSAT scenario, but especially significant for current day QSAT solvers has been the extension of the DPLL procedure for propositional logic [Davis et al., 1962] to quantified boolean logic in [Cadoli et al., 1998] and, to a lesser degree, initial developments in so called "variable elimination approach" approach based solvers (some early examples of this approach are presented in, for example, [McMillan, 1993, Biere, 2004]).

The increasing number of uses of QSAT solvers for solving computational problems, witnesses to the perceived adequacy of this approach (see [Giunchiglia et al., 2009] for various examples). Planning [Rintanen, 1999a, Ferraris and Giunchiglia, 2000] and formal methods for determining correctness of software systems [Bryant et al., 2003, Mneimneh and Sakallah, 2003, Ling et al., 2005, Benedetti and Mangassarian, 2008, Giunchiglia et al., 2007] are notable examples of areas where QSAT solvers have found their way into practice. Moreover, reductions of many reasoning problems in different areas including knowledge representation and reasoning (see, for example, [Egly et al., 2000] for reductions of autoepistemic, default logic, disjunctive logic programming, and circumscription problems) into QSAT have also been proposed in the literature.

All the above should underpin our reasons for choosing quantified boolean logic as target formalism in which to translate reasoning for ADFs. In the first place, as has already been noted above, this provides a uniform axiomatization of reasoning on ADFs into a relatively well established formalism of formal logic. In the second place, quantified boolean logic also has just the "right level of expressivity", given known results about complexity of reasoning for ADFs and the connections between quantified boolean logic and the polynomial hierarchy detailed above. Finally, the mentioned advances in the development of QSAT solvers makes this undertaking worthwile from a practical perspective as well as a theoretical one.

On the theoretical side, our work continues the line of of study that has been initiated for Dung style argumentation frameworks in [Egly and Woltran, 2006] and [Arieli and Caminada, 2012, Arieli and Caminada, 2013] which we have already made reference to previously. In both of these works, reductions of the problems of evaluating Dung style argumentation frameworks into the setting of quantified boolean logic are given. The more recent work by Arieli and Caminada in particular, which is based on a so called "labelling approach" to defining the semantics of AFs which presents some parallels with the approach followed in [Brewka et al., 2013] to define the semantics for ADFs, has been particulary influential for the approach we follow in this work to give encodings for the reasoning problems defined for ADFs.

We briefly summarise the main contributions of our work:

1. We provide complexity-sensitive encodings of some of the main reasoning problems defined for ADFs (evaluation, existence, non-trivial existence, credulous acceptance, skeptical acceptance) with respect to some of the major semantics (three and two valued models, admissible, complete, preferred, grounded) into two of the most significant problems for quantified boolean logic (QSAT and model enumeration).

2. As a by-product of the work described in Item 1, we also prove the NP-completeness of non-trivial existence and credulous acceptance with respect to three valued models for ADFs.

3. We present a prototype software system for reasoning on ADFs based on the encodings we provide in this work and report on preliminary experiments.

Regarding the structure of our work, in Chapter 2 we present the background theory on which it is based, mainly propositional and quantified boolean logic, abstract dialectical frameworks, as well as relevant concepts and results from complexity theory. In Chapter 3 we present the encodings of the reasoning tasks we consider in this work with respect to ADFs, together with proofs of correctness and analysis to determine the adequacy of the encodings with respect to their complexity. As part of the latter analysis we prove NP-completeness of non-trivial existence and credulous acceptance with respect to three valued models for ADFs in Section 3.2. Finally, in Chapter 4 we present the before-mentioned prototype system for reasoning on ADFs via the QSAT-solver `DepQBF` and report on preliminary experiments, not before first setting our system in context by reviewing developments in logic-based reductions and implementations of abstract argumentation as well as some of the main theoretical and practical advances in QSAT existing to this date.

CHAPTER 2

Background

Throughout this work we assume only familiarity with basic concepts from set theory and algorithmics, introducing most of the remaining background knowledge required to grasp our work in this chapter. Regarding the notation for set theory and algorithms, we make use of standard notation of set theory (e.g. $\setminus$, $\cup$, $\uplus$, $\cap$, $\times$ for difference, union, disjoint union, intersection, and cartesian product of sets, and $|S|$ for the cardinality of a set S) as well as familiar pseudo-code notation for presenting algorithms.

In Sections 2.1 and 2.2 we introduce the syntax, semantics, as well as certain useful syntactical normal forms for classical propositional and quantified boolean logic respectively. In Section 2.3 we introduce ADFs as well as (Dung style) AFs as a special case of ADFs. Most of this section is devoted to defining the semantics for ADFs (and AFs) that are relevant for this work. Finally, in Section 2.4 we introduce the main decision problems for which we provide encodings in this work, give a brief overview of complexity theory, and summarise complexity-theoretic results that are important for the present work.

2.1 Propositional logic

From the perspective of knowledge representation and reasoning, propositional logic is mainly a formalism for representing and reasoning about propositions, statements or sentences and combinations thereof. Classical (truth valued) propositional logic in particular assumes that propositions, statements or sentences are atomic and can be either true or false (but not both) and allows only truth functional combinations of these, i.e. only those combinations whose truth value depends on the truth values of the simpler statements which make them up [Klement, 2013]. In the rest of this work, we refer to "classical propositional logic" simply as "propositional logic".

Although we do not delve on this point further here, what makes propositional logic especially interesting for the purposes of knowledge representation and reasoning is in the first place that many hard computational problems can be encoded in an efficient manner as reasoning problems in this logic (see, for example, [Kautz and Selman, 1992, Biere, 2009, Kroening, 2009, Rintanen, 2009, Zhang, 2009]). In fact, the satisfiability problem for propositional logic that we introduce in this section is significant for complexity theory because of its connection to the so called “complexity class NP” [Garey and Johnson, 1979]. Thanks to [Cook, 1971] the satisfiability problem for propositional logic can be considered the “prototypical problem” for the class NP in the sense that this class can be defined as the set of problems that can be expressed in terms of the satisfiability problem in an efficient manner. Equally significant is that, as has been mentioned in the introduction of this work, the recent two decades have given rise to increasingly efficient automatic reasoning systems for this logic [Järvisalo and Gelder, 2013]. Many of these at their core nevertheless still rely on the DPLL backtracking decision procedure for this logic introduced in the early 1960s [Davis et al., 1962].

The definitions and results we present here can be found in most textbooks on logic. For a more detailed introduction to propositional logic from a formal and computational perspective, we refer to e.g. [Büning and Letterman, 1999, Hölldobler, 2011].

Syntax

In the following we present the syntax of propositional logic. To start, we define the alphabet underlying this formalism:

Definition 2.1.1. The *alphabet* of the language of propositional logic consists of the following symbols:

- A countable set of propositional *variables* or *atoms* $\mathcal{P} = \{p_1, p_2, p_3, ...\}$.
- The *logical constants* $\top$ (*truth constant*), $\bot$ (*falsity constant*).
- The *logical connectives* $\neg$ (*negation sign*), $\wedge$ (*conjunction sign*), and $\vee$ (*disjunction sign*).
- The auxiliary symbols “(“ and “)”.

In the metalanguage, we will often also use symbols p, q, $\ldots$ (each of them possibly with subscripts, superscripts or primed) to refer to propositional variables. We also consider $\mathcal{P}$ to be suitably large for our purposes. More precisely, for any $p \in \mathcal{P}$ we will assume that also $p_b^t \in \mathcal{P}$ where b and t are arbitrary (including none) subscripts and superscripts respectively.

Having defined the alphabet of propositional logic, the formulas of propositional logic can also be defined.

Definition 2.1.2. Given the alphabet as in Definition 2.1.1, the set of *formulas* of propositional logic is defined inductively as follows:

- Any propositional variable $p \in \mathcal{P}$ is a formula.
- Any of the logical constants $\top$ and $\bot$ is a formula.
- If ϕ and ψ are formulas, then so is $(\neg\phi)$, $(\phi \wedge \psi)$, and $(\phi \vee \psi)$.
- Nothing else is a formula.

Propositional variables and negated propositional variables are often referred to as *literals*, the first being called *positive* and the second *negative*. For a literal l, we define $\bar{l} = \neg p$ if $l = p$ and $\bar{l} = p$ if $l = \neg p$ for some $p \in \mathcal{P}$.

Given a propositional formula ϕ, the set of subformulas of ϕ can be extracted based on the structure of ϕ:

Definition 2.1.3. The *set of subformulas* of a formula ϕ, SUBFORMULAS(ϕ), is defined as follows:

- If ϕ is a propositional variable $p \in \mathcal{P}$, $\top$ or $\bot$ then SUBFORMULAS(ϕ) $= \{\phi\}$.
- If ϕ is a formula of the form $(\neg\psi)$ then SUBFORMULAS(ϕ) $= \{\phi\} \cup$ SUBFORMULAS(ψ).
- If ϕ is a formula of the form $(\psi \wedge \rho)$ or $(\psi \vee \rho)$, then SUBFORMULAS(ϕ) $= \{\phi\} \cup$ SUBFORMULAS(ψ) $\cup$ SUBFORMULAS(ρ).

Finally, a formula ψ is a *subformula* of a formula ϕ and, hence, *occurs* in ϕ if $\psi \in$ SUBFORMULAS(ϕ).

Clearly, a formula ψ can occur more than once in another formula ϕ. In this work we assume it to be clear from context to which occurence we are referring to and do not define this concept formally. Also, for the case of literals, it is convenient to use the notation $|l|$ to denote the propositional variable occuring in a literal l.

One can define convenient abbreviations for some of the combinations of propositional formulas that can be constructed using the connectives in the alphabet using new *defined connectives* such as $\rightarrow$ (*implication sign*), $\leftrightarrow$ (*biconditional sign*), $\oplus$ (*exclusive disjunction sign*). If ϕ and ψ are propositional formulas, the definition of these is as follows:

$$\phi \rightarrow \psi := ((\neg\phi) \vee \psi)$$

$$\phi \leftrightarrow \psi := ((\phi \rightarrow \psi) \wedge (\psi \rightarrow \phi))$$

$$\phi \oplus \psi := ((\phi \vee \psi) \wedge (\neg(\phi \wedge \psi)))$$

For purposes of clarity, we will in this work at times use "[" and "]" instead of "(" and ")" when writing formulas. Also, we will at times omit parentheses, for which we introduce the following ranking (in increasing order) to the connectives we have introduced above:

$$\neg, \wedge, \vee, \oplus, \rightarrow, \leftrightarrow$$

When parentheses are omited, parentheses should be read into the formula according to this ranking (i.e. connectives lower in the ranking "bind stronger") to avoid ambiguity. We will further simplify our rendering of formulas by assuming that binary connectives associate to the left (so, for example, $\phi_1 \wedge \phi_2 \wedge \phi_3$ should be read as $((\phi_1 \wedge \phi_2) \wedge \phi_3)$).

Other useful abbreviations we will use in this work to simplify formulas when writing them out are those of *n-ary conjunctions* or *disjunctions*. Given a set $F = \{\phi_1, \phi_2, ..., \phi_n\}$ of formulas, these abbreviations are defined as follows:

- $\bigwedge_{\phi \in F} = \bigwedge_{i=1}^{n} \phi_i := \phi_1 \wedge \phi_2 \wedge ... \wedge \phi_n$ (n-ary conjunction)
- $\bigvee_{\phi \in F} = \bigvee_{i=1}^{n} \phi_i := \phi_1 \vee \phi_2 \vee ... \vee \phi_n$ (n-ary disjunction)

If $F = \emptyset$, we stipulate the above abbreviations to simplify to $\top$ for the *empty conjunction* and $\bot$ for the *empty disjunction*.

Semantics

The semantics of propositional formulas are based on the notion of a valuation or assignment:

Definition 2.1.4. A *valuation* or *assignment* v (over $\mathcal{P}$) is a function from $\mathcal{P}$ to $\{1, 0\}$. A valuation is *complete* if it is a total function, otherwise it is *partial*.

Except if explicitly stated otherwise, when we refer to valuations in this work we assume that the valuation in question is complete. A valuation can be transformed into a new one by changing the assignment given to some $p \in \mathcal{P}$:

Definition 2.1.5. Given a valuation v, $v[p/x]$ where $p \in \mathcal{P}$ and $x \in \{0, 1\}$ denotes the valuation v' defined as:

- $v'(p) = x$.
- $v'(q) = v(q)$ if $q \in \mathcal{P}$ and $q \neq p$.

If P is a sequence of distinct propositional variables $\{p_1, p_2, .., p_n\}$ and Y a sequence of values in $\{0, 1\}$, $\{x_1, x_2, .., x_n\}$, $(|P| = |Y|)$, then $v[P/Y]$ denotes $v[p_1/x_1][p_2/x_2]...[p_n/x_n]$.

A valuation can also be restricted to a given set of propositional variables:

Definition 2.1.6. Let v be a valuation and P a set of propositional variables. $v|_\mathrm{P}$ denotes the *restriction of v to P* which is the partial valuation with domain P and such that $v|_\mathrm{P}(p) = v(p)$ for every $p \in$ P.

Valuations can be extended to arbitrary propositional formulas as follows:

Definition 2.1.7. The value of a propositional formula under a valuation v, is a mapping v^* from propositional formulas to $\{1, 0\}$ defined inductively as follows:

- $v^*(p) = v(p)$ for $p \in \mathcal{P}$.
- $v^*(\top) = 1$.
- $v^*(\bot) = 0$.

- $v^*(\neg\phi) = 1$ if $v^*(\phi) = 0$, otherwise $v^*(\neg\phi) = 1$ for any formula ϕ.
- $v^*(\phi \wedge \psi) = 1$ if $v^*(\phi) = 1$ and $v^*(\psi) = 1$, otherwise $v^*(\phi \wedge \psi) = 0$ for any formulas ϕ and ψ.
- $v^*(\phi \vee \psi) = 1$ if $v^*(\phi) = 1$ or $v^*(\psi) = 1$ (or both), otherwise $v^*(\phi \vee \psi) = 0$ for any formulas ϕ and ψ.

By abuse of notation, we will omit the $*$ when referring to the extension of a valuation to arbitrary propositional formulas in the rest of this work.

The following proposition is a straightforward consequence of the semantics of propositional logic:

Proposition 2.1.1. Let $\phi_1, .., \phi_n$ be arbitrary formulas. Then

- $v(\bigwedge_{i=1}^{n} \phi_i) = 1$ if and only if $v(\phi_i) = 1$ for each $0 \leq i \leq n$.
- $v(\bigvee_{i=1}^{n} \phi_i) = 1$ if and only if $v(\phi_i) = 1$ for some $0 \leq i \leq n$.

The notion of valuation allows us to define some of the central semantical concepts of logic for propositional logic:

Definition 2.1.8. Let ϕ be a propositional formula.

- A valuation v is a *model* for ϕ if $v(\phi) = 1$. This is written as $v \models \phi$.
- ϕ is *satisfiable* if there is a model for ϕ, ϕ is *unsatisfiable* otherwise.
- ϕ is *valid* (or a *tautology*) if all valuations are models of ϕ.
- ϕ is a *logical consequence* (or just *consequence*) of a set of formulas Γ, denoted $\Gamma \models \phi$, if all valuations that are a model of each of the formulas in Γ are a model of ϕ.

Two formulas are equivalent from the perspective of classical propositional logic if their truth values are identical under all possible valuations to the propositional variables ("principle of extensionality"):

Definition 2.1.9. Let ϕ and ψ be propositional formulas. ϕ and ψ are *equivalent* if for all valuations v over $\mathcal{P}$, $v(\phi) = v(\psi)$. When two formulas ϕ and ψ are equivalent, this is denoted as $\phi \equiv \psi$.

A weaker concept than that of equivalence is that of two formulas being "equisatisfiable":

Definition 2.1.10. Let ϕ and ψ be propositional formulas. ϕ and ψ are *equisatisfiable* if ϕ is satisfiable if and only if ψ is.

As a result of the truth functional nature of classical propositional logic, equivalent formulas can be substituted in a formula without changing the valuation of the formula in question. This is captured by the so called "*Replacement Theorem*":

Theorem 2.1.1. Let ϕ be a formula in which ψ occurs and ρ be a formula such that $\psi \equiv \rho$. If ϕ' is the formula that results from the replacement of each occurrance of ψ for ρ, then $\phi \equiv \phi'$.

Finally, the following are some important semantic equivalences used in this work:

Theorem 2.1.2. Let ϕ, ψ, and ρ be arbitrary propositional formulas. Then

- $(\phi \wedge \psi) \equiv (\psi \wedge \phi)$ and $(\phi \vee \psi) \equiv (\psi \vee \phi)$ (*Commutativity laws*)
- $\neg(\phi \wedge \psi) \equiv \neg\phi \vee \neg\psi$ and $\neg(\phi \vee \psi) \equiv \neg\phi \wedge \neg\psi$ (*De Morgan's laws*)
- $\neg\neg\phi \equiv \phi$ (*Principle of double negation*)
- $\phi \wedge (\psi \vee \rho) \equiv (\phi \wedge \psi) \vee (\phi \wedge \rho)$ and $\phi \vee (\psi \wedge \rho) \equiv (\phi \vee \psi) \wedge (\phi \vee \rho)$ (*Distributivity laws*)

Conjunctive normal form

For computational purposes it is often useful to concentrate on formulas with restricted syntactic structure. Especially useful is if these sets of formulas capture all possible formulas from a semantic point of view. In this work the so called "conjunctive normal form" which has precisely this property plays an important role.

Definition 2.1.11. A *clause* is a disjunction of literals, i.e. a formula of the form $C = l_1 \vee l_2 \vee ... \vee l_n$ where each of the $l_i s$ for $1 \leq i \leq n$ is a literal. A formula is in *conjunctive normal form* (CNF) if it is a conjunction of clauses, i.e. a formula of the form $C_1 \wedge C_2 \wedge ... \wedge C_m$ where each of the $C_i s$ for $1 \leq i \leq m$ is a clause.

Note that checking that a formula in CNF is valid reduces to checking that in every clause of the formula some variable as well as its negation occur. The following theorem expresses that no power of expressivity is lost by restricting our attention to formulas in CNF:

Theorem 2.1.3. For each propositional formula ϕ there exists a formula ϕ' in CNF such that $\phi \equiv \phi'$.

The following is a function transforming an arbitrary formula ϕ into CNF [Huth and Ryan, 2004]. It assumes that the formula ϕ is in *negation normal form*, i.e. that no defined connectives appear in the formula and that negations appear only in front of variables. A formula is easily transformed into an equivalent formula in negation normal formula in a recursive manner using the definitions of $\rightarrow$, $\leftrightarrow$, $\oplus$ as well as De Morgan's laws and the law of double negation.

```
function CNF($\phi$):
  if $\phi$ is a literal, return $\phi$
  if $\phi$ is $\psi \wedge \rho$ return CNF($\psi$) $\wedge$ CNF($\rho$)
  if $\phi$ is $\psi \vee \rho$ return DISTRIB(CNF($\psi$) $\vee$ CNF($\rho$))
```

The function DISTRIB repeatedly makes use of the second distributivity law (and commutativity of $\vee$) to transform the formula:

```
function DISTRIB($\phi \vee \psi$):
  if $\phi$ is $\phi_1 \wedge \phi_2$ return DISTRIB($\phi_1, \psi$) $\wedge$ DISTRIB($\phi_2, \psi$)
  if $\psi$ is $\psi_1 \wedge \psi_2$ return DISTRIB($\psi_1, \phi$) $\wedge$ DISTRIB($\psi_2, \phi$)
  otherwise (no conjunctions) return $\phi \vee \psi$
```

A constructive proof of Theorem 2.1.3 can be given relying on the function CNF as well as the definitions of the defined connectives, Theorem 2.1.2, and the Replacement Theorem.

Although the function CNF as defined above is correct, the use of the function DISTRIB (and elimination of $\leftrightarrow$) can lead to an exponential explosion in the size of the resulting formula. A linear transformation to CNF form is the so called "Tseitin encoding" first defined in [Tseitin, 1968] which works by replacing subformulas with new variables and then adding clauses to the original formula specifying the relationship between the newly introduced variables and the subformulas. This results in an *equisatisfiable* formula, although when satisfiability of the original formula is the issue of concern this is obviously not a disadvantage.

2.2 Quantified boolean formulas

In the following we formally introduce quantified boolean formulas, or more accurately, quantified boolean *logic* as an extension to propositional logic. Existential ($\exists$) and universal ($\forall$) quantification over propositional variables are added to the alphabet of propositional logic and from a semantic perspective this will, as we show in this section, allow expressing properties about propositional valuations themselves.

Quantified boolean logic is a conservative extension of propositional logic in the sense that the satisfiability problem for quantified boolean formulas can be translated into the satisfiability problem for propositional formulas. On the other hand, this translation may lead to an exponential blow up in the size of the target formula and the power of quantified boolean formulas lies in their potential to express complex properties that can be specified in propositional logic in a more succinct manner. Quantification over propositional variables, hence, allows for more "natural" encodings of some computational problems as well as a potential reduction of memory space needed to express these.

Just as the satisfiability problem for propositional logic can be considered the prototypical problem of the class of problems NP, thanks to [Stockmeyer and Meyer, 1973], the satisfiability problem for QBFs can be considered analogously for the complexity class PSPACE. Assuming the problems in NP are a proper subset of those in PSPACE, this expresses the space vs. time trade-off inherent in the choice of use of quantified boolean logic vs. propositional logic for modelling and reasoning purposes. On the other hand, while problems in PSPACE (like model checking for linear temporal logic) have, in the past, been translated into propositional satisfiability problems, recent increase in the power of QBF solvers which we refer to with a little more detail in Section 4.2, provides an additional argument in favour of the use of quantified boolean logic as a modelling framework when dealing with problems in PSPACE.

See the introduction (Chapter 1) of this work for pointers to uses of quantified boolean logic for solving computational problems. A short but useful overview of the theory of quantified boolean logic is, for example, provided in [Büning and Bubeck, 2009]. We have also used [Woltran, 2003] as a basis for this section, which presents quantified boolean logic from the perspective of the use of this logic to provide encodings of reasoning tasks from the area of knowledge representation and reasoning.

Syntax

As indicated above, the *alphabet* of the language of quantified boolean logic consists of the symbols of the language of propositional logic as in Definition 2.1.1 and the symbols $\forall$ and $\exists$. Quantified boolean formulas can then be defined analogously to the definition of formulas of propositional logic:

Definition 2.2.1. Given the alphabet of quantified boolean logic, the set of *quantified boolean formulas* (QBFs) is defined inductively as follows:

- Any propositional variable $p \in \mathcal{P}$ is a QBF.
- Any of the logical constants $\top$ and $\bot$ is a QBF.
- If ϕ and ψ are QBFs, then so is $(\neg\phi)$, $(\phi \wedge \psi)$, and $(\phi \vee \psi)$.
- If $p \in \mathcal{P}$ is a propositional variable and ϕ is a QBF, then $(\exists p\phi)$ and $(\forall p\phi)$ are QBFs.
- Nothing else is a QBF.

The definition of subformulas can also be extended to QBFs:

Definition 2.2.2. The *set of subformulas* of a QBF ϕ, SUBFORMULAS(ϕ), is defined as follows:

- If ϕ is a propositional variable $p \in \mathcal{P}$, $\top$ or $\bot$ then SUBFORMULAS$(\phi) = \{\phi\}$.
- If ϕ is a QBF of the form $(\neg\psi)$, $(\forall p\psi)$, or $(\exists p\psi)$ then SUBFORMULAS$(\phi) = \{\phi\} \cup$ SUBFORMULAS(ψ).
- If ϕ is a QBF of the form $(\psi \wedge \rho)$ or $(\psi \vee \rho)$, then SUBFORMULAS$(\phi) = \{\phi\} \cup$ SUBFORMULAS$(\psi) \cup$ SUBFORMULAS(ρ).

Finally, a QBF ψ is a *subformula* of a QBF ϕ and, hence, *occurs* in ϕ if $\psi \in$ SUBFORMULAS(ϕ).

When it is clear from the context that we are referring to QBFs and not propositional formulas, we will often refer to QBFs as formulas in this work.

All of the syntactic notions and abbreviations of propositional logic are extended to quantified boolean logic in a straightforward manner (i.e. by replacing “formulas” for “QBFs” in the places where we introduced these notions and abbreviations).

In order to simplify QBFs we also stipulate that the quantifiers $\exists$ and $\forall$ have the same ranking as the symbol $\neg$ when parentheses are to be read into formulas where these are ommited. In addition to the abbreviations introduced when considering the syntax of propositional logic, we also introduce syntactic sugar for writing out formulas with repeated occurances of quantifiers of the same type. Specifically, if $P = \{p_1, p_2, ..., p_n\}$ is a set of propositional variables and ϕ a QBF, then $QP\phi$ and $Qp_1p_2...p_n\phi$ are to be read as $(Qp_1(Qp_2(...(Qp_n(\phi)))))$ for any $Q \in \{\exists, \forall\}$. In particular, if $P = \emptyset$ then $QP\phi$ is to be read as ϕ. We call successive quantifiers of the same kind occuring in a certain formula a *quantifier block*.

Important syntactical notions regarding QBFs are the scope in which a quantifier is applied and whether a variable appears bound by a quantifier or not in such a formula.

Definition 2.2.3. The *scope* of a quantifier $Q \in \{\forall, \exists\}$ in a QBF of the form $Qp\phi$ is the QBF ϕ. An occurence of a variable p in a QBF ϕ is *free* if it does not occur in the scope of a quantifier in the QBF, otherwise the occurence of p is *bound*. If a QBF ϕ contains no free variable occurences, then ϕ is *closed*, otherwise ϕ is *open*. FREE(ϕ) denotes the set of free variables of a QBF ϕ.

With these notions in hand, substitution of formulas for variables occurring in a QBF can be defined:

Definition 2.2.4. Let ϕ be a QBF, $\{\psi_1, ..., \psi_n\}$ a set of QBFs such that none of the propositional variables in $P = \{p_1, .., p_n\}$ occurs *free* in any of the ψ_is for $1 \leq i \leq n$. Then, $\phi[p_1/\psi_1, ..., p_n/\psi_n]$ denotes the QBF which results by *uniform substitution* of all free occurences of the variables p_i in ϕ by the corresponding ψ_i for $1 \leq i \leq n$.

Semantics

The semantics of quantified boolean logic is also based on the notion of valuation defined for propositional logic. Valuations can then be extended to arbitrary quantified boolean formulas as for propositional logic, but adding conditions for quantified formulas:

Definition 2.2.5. The value of a QBF under a valuation v, is a mapping v^* from QBFs to $\{1, 0\}$ defined inductively as follows:

- $v^*(p) = v(p)$ for $p \in \mathcal{P}$.
- $v^*(\top) = 1$.
- $v^*(\bot) = 0$.
- $v^*(\neg\phi) = 1$ if $v^*(\phi) = 0$, otherwise $v^*(\neg\phi) = 1$ for any QBF ϕ.
- $v^*(\phi \wedge \psi) = 1$ if $v^*(\phi) = 1$ and $v^*(\psi) = 1$, otherwise $v^*(\phi \wedge \psi) = 0$ for any QBFs ϕ and ψ.

- $v^*(\phi \vee \psi) = 1$ if $v^*(\phi) = 1$ or $v^*(\psi) = 1$ (or both), otherwise $v^*(\phi \vee \psi) = 0$ for any QBFs ϕ and ψ.
- $v^*(\exists p\phi) = 1$ if $v^*(\phi[p/\top]) = 1$ or $v^*(\phi[p/\bot]) = 1$, otherwise $v^*(\exists p\phi) = 0$ for any QBF ϕ and $p \in \mathcal{P}$.
- $v^*(\forall p\phi) = 1$ if $v^*(\phi[p/\top]) = 1$ and $v^*(\phi[p/\bot]) = 1$, otherwise $v^*(\forall p\phi) = 0$ for any QBF ϕ and $p \in \mathcal{P}$.

As for propositional logic, we will ommit the $*$ when referring to the extension of a valuation to arbitrary QBFs in the rest of this work.

The notions of *model* of a QBF, a QBF being *satisfiable*, *unsatisfiable* or *valid* as well as a QBF being a *logical consequence* of a set of QBFs are then defined as for propositional logic but referring to QBFs and using the semantical definitions for QBFs. From the definitions of the semantics of the quantifiers it can be inferred that for closed QBFs, the concepts of satisfiability and validity coincide:

Proposition 2.2.1. Let ϕ be an arbitrary closed QBF. Then ϕ is satisfiable if and only if ϕ is valid.

Hence, a closed QBF can be considered *true* when satisfiable (or valid) and *false* otherwise.

With the semantical notions for QBFs in place it can be seen how quantified boolean logic can be used to express properties about (partial) propositional valuations. For example, the fact that for all partial valuations to the variable p, there exists a partial valuation to the variable q which is dual to p, can be expressed as follows:

$$\forall p \exists q(p \leftrightarrow \neg q)$$

which can be easily seen to be a true formula according to the semantics. On the other hand, a slight syntactic change in this formula leads to a false formula:

$$\exists p \forall q(p \leftrightarrow \neg q)$$

Satisfiability and validity of propositional formulas can be expressed via QBFs as follows:

Proposition 2.2.2. Let ϕ be an arbitrary propositional formula with P being all variables that appear in ϕ. Then ϕ is satisfiable if and only if $\exists P\phi$ is true and ϕ is valid if and only if $\forall P\phi$ is true.

In fact, the proposition above continues to hold if ϕ is a quantified boolean formula and P are the free variables appearing in ϕ. On the other hand, as mentioned in the introduction to this section, the problem of satisfiability of quantified boolean formulas can be translated into that of satisfiability of propositional formulas. This translation, which is sometimes referred to as the “Shannon expansion”, is straightforward from the semantics of QBFs and the following equivalences:

$$\exists p\phi \equiv \phi[p/\bot] \vee \phi[p/\top]$$

$\forall p\phi \equiv \phi[p/\bot] \wedge \phi[p/\top]$

Because satisfiability of a QBF with free variables can be expressed via a closed QBF, it suffices to consider closed QBFs for the translation.

Theorem 2.2.1.

Let ϕ be a closed QBF and $S(\phi)$ its Shannon expansion. Then ϕ is true if and only if $S(\phi)$ is satisfiable.

On the other hand, it should be clear that this translation can lead to an exponential increase in the size of the target formula.

Given the truth functional nature of propositional logic, Proposition 2.1.1 continues to hold when "formula" is considered to refer to QBFs. Also, the notion of semantical equivalence for QBFs is also defined as for propositional logic. All the semantical equivalences given for propositional formulas in Theorem 2.1.2 also hold for QBFs and the following are semantic equivalences specific to quantified boolean logic which will be used in this work:

Theorem 2.2.2. Let ϕ, ψ be arbitrary QBFs with $p \in \mathcal{P}$ not occurring free in ψ. Then

- $(\neg\exists p\phi) \equiv (\forall p\neg\phi)$
- $(\neg\forall p\phi) \equiv (\exists p\neg\phi)$
- $(Qp\phi \wedge \psi) \equiv Qp(\phi \wedge \psi)$ for any $Q \in \exists, \forall$
- $(\phi \wedge Qp\psi) \equiv Qp(\phi \wedge \psi)$ for any $Q \in \exists, \forall$
- $(Qp\phi \vee \psi) \equiv Qp(\phi \vee \psi)$ for any $Q \in \exists, \forall$
- $(\phi \vee Qp\psi) \equiv Qp(\phi \vee \psi)$ for any $Q \in \exists, \forall$

Prenex conjunctive normal form

Just as in the case of propositional logic, for computational purposes it is often useful to restrict ones attention to certain normal forms when writing QBFs. The "prenex conjunctive normal form" (PCNF) plays an important role in the present work. First we introduce the following equally useful syntactic restriction to QBFS:

Definition 2.2.6. A QBF ϕ is *standarized apart* if the following properties hold:

- No variable ocurring in ϕ occurs both free and bound.
- For each Q_1,$Q_2 \in \{\forall, \exists\}$, if Q_1p and Q_2q are two distinct occurences of quantifiers in ϕ then p and q are distinct variables.
- For each subformula $Qp\psi$ with $Q \in \{\forall, \exists\}$ ocurring in ϕ, p is a free variable in ψ.

An arbitrary QBF ϕ can be transformed in linear time into a ϕ' which is equivalent to ϕ and standarized apart by renaming variables which do not satisfy the first two conditions in the definition and removing Qp for $Q \in \{\exists, \forall\}$ in $Qp\psi$ when p does not occur freely in ψ. That the resulting formula ϕ' is equivalent to ϕ is based on the following observations:

Lemma 2.2.1. For a propositional variable p not occuring freely in a QBF ψ and $Q \in \{\forall, \exists\}$,

$$(Qq\psi) \equiv (Qp\psi[q/p])$$

$$(Qp\psi) \equiv (\psi)$$

Now the definition of the prenex conjunctive normal form can be given as follows:

Definition 2.2.7. A QBF ϕ is in *prenex normal form* if it is standarized apart and it is of the form

$$Q_1P_1Q_2P_2...Q_nP_n\psi$$

where ψ is a propositional formula, $Q_i \in \{\forall, \exists\}$ for $1 \leq i \leq n$, and the P_is are (mutually disjoint) sets of propositional variables. ψ is called the *matrix* of ϕ and $Q_1P_1Q_2P_2...Q_nP_n$ is the *prefix* of ϕ. Finally, ϕ is in *prenex conjunctive normal form* if it is in prenex normal form and its matrix is in conjunctive normal form.

A standard procedure for transforming a QBF ϕ into an equivalent formula ϕ' in PCNF is by first standarizing apart. Then, defined connectives are eliminated in terms of their definitions and subformulas of the form $\neg\forall p\psi$ are transformed into $\exists p\neg\psi$ and $\neg\exists p\psi$ into $\forall p\neg\psi$ in recursive manner. Subsequently, quantifiers are "pulled out" by using the last four equivalences in Theorem 2.2.2 (from left to right). Finally, the matrix of the resulting formula in prenex normal form can be transformed into CNF by using the procedure described in the section on propositional logic. A constructive proof of the following theorem can rely on this procedure, the definitions of the defined connectives, Theorem 2.2.2 and the Replacement Theorem for QBFs.

Theorem 2.2.3. For any arbitrary QBF ϕ there exists an equivalent QBF ϕ' in PCNF such that $\phi \equiv \phi'$.

The procedure to transform a QBF into PCNF can lead to an exponential explosion because of the transformation of the matrix into CNF (as well as replacing $\leftrightarrow$ for its definition). The "Tseitin encoding" described in the section on CNF can nevertheless be adapted to QBFs to achieve a linear transformation resulting in an equivalent (since closed) formula.

For a QBF in PCNF ϕ, a literal l occuring in ϕ is *existential* if $\exists|l|$ belongs to the prefix, and it is *universal* otherwise. The *level* of a literal (or variable) l in a QBF in PCNF $Q_1P_1Q_2P_2...Q_nP_n\psi$ is 1 + the number of expressions $Q_jP_jQ_{j+1}P_{j+1}$ in the prefix with $j \geq i$ and $Q_j \neq Q_{j+1}$.

Finally, we make note of some equivalence preserving simplifications of QBFs in PCNF and checks that are useful when determining satisfiability that will be used when we survey some of the existing approaches for determining satisfiability of QBFs later in this work (Section 4.2). To ease the presentation of these observations, we first introduce the operation of *propagation on a literal* l in ϕ which we denote ϕ_l and will also need in later sections of this work:

Definition 2.2.8. If l is a literal with $|l| = p$, then ϕ_l is the QBF resulting from l by removing all clauses in which l occurs in the matrix of ϕ, $\bar{l}$ from all other clauses, and Qp from the prefix of the QBF. If μ is a sequence of literals $l_1, l_2, ..., l_m$, then ϕ_μ is defined to be $(...((\phi_{l_1})_{l_2}...)_{l_m}$.

Now, to the above mentioned simplifications and checks [Büning and Bubeck, 2009]:

Lemma 2.2.2. In all the following, ϕ is a QBF in PCNF.

1. (**Elimination of tautological clauses**) A *tautological clause* in the matrix of a QBF in PCNF is a clause that contains both p and $\neg p$ for some $p \in \mathcal{P}$. If ϕ is not a tautology, such clauses can be removed from the matrix of ϕ preserving equivalence.

2. (**Trivial falsity on contradictory clauses**) If a *contradictory* clause, i.e. a non-tautological clause with no existential literals, is a subformula of ϕ, then the formula is false.

3. (**Universal reduction**) Assume a non-tautological clause C of the form $(\psi \vee l)$ occurs in ϕ, where l is a universal literal and ψ a sub-clause. If there is no existential literal in ψ, or if the level of every existential literal in ψ is lower than that of l, the literal l can be deleted from the clause C in ϕ while preserving equivalence.

4. (**Unit propagation**) A literal l is *unit* in ϕ if l is existential and it occurs in a clause in which each other literal is universal and has a higher level than l. If l is unit in ϕ, then $\phi \equiv \phi_l$.

5. (**Pure or monotone literal detection**) A literal l is *pure* or *monotone* in ϕ if either l is existential, l occurs in ϕ and $\bar{l}$ does not or l is universal, $\bar{l}$ occurs in ϕ and l does not. If l is pure or monotone in ϕ, then $\phi \equiv \phi_l$.

2.3 Abstract dialectical frameworks

Abstract dialectical frameworks (ADFs) are a relatively new formalism developed for abstract argumentation. As has been noted in the introduction to this work (Section 1), from the perspective of argumentation, the central concern of abstract argumentation is determining the acceptance of (sets of) arguments based on an abstract representation of the relationships between these arguments, i.e. more or less disregarding the "internal structure" of the arguments.

The role of abstract argumentation can be appreciated better by considering its role in a relatively simple framework for formally modelling what has been called the "argumentation process" in the introduction to this work, which is relatively influential among researchers in argumentation from computer science and artificial intelligence [Caminada and Amgoud, 2007]. This model considers the argumentation process to be decomposable into several relatively independent steps. It pertains especially to "monological argumentation" [Besnard and Hunter, 2008], where a a single agent collates information from possibly heterogeneous sources to construct arguments for and against a particular conclusion and is, therefore, also particularly relevant to modelling non-monotonic reasoning as a form of argumentation. It can, nevertheless, also be of use beyond this scenario as, for example, when used for the reconstruction and analysis of an argumentation between several agents.

In this model one starts with a knowledge base KB, which will usually be a set of formulas of some logic $\mathcal{L}$, for example, propositional logic. This set may be inconsistent according to the logic in question. The first step is to construct arguments based on the knowledge base, where an argument will usually be defined in terms of the logical consequence relation of the logic being used. For example, in [Besnard and Hunter, 2008] a notion of an argument based on a KB consisting of propositions of propositional logic is defined as a pair (Φ, α) such that $\Phi \subset$ KB is consistent, $\Phi \models \alpha$, and for no $\Psi \subset \Phi$, $\Psi \models \alpha$. Having constructed the arguments, the relationships between the arguments can be determined based on some notion once again defined in terms of the logical theory in question. Continuing with our example, a basic type of conflict between arguments (Φ, α) and (Φ', α') is when $\Phi \cup \alpha' \models \bot$. In this case one could see (Φ', α') as "attacking" (Φ, α).

Having identified the relationships among the arguments allows for abstracting away from the internal structure of the arguments to construct some abstract representation of this relationship. This step is where abstract argumentation comes into play. For example, in the first model of abstract argumentation proposed by Dung [Dung, 1995], argumentation is modelled as an abstract framework (AF): a directed graph where nodes represent arguments and links correspond to an argument attacking another.

Evaluation of arguments can be described informally to be to determine whether an argument has some way of surviving the attacks it receives. Alternatively, it can be of interest to determine which sets of arguments can "survive together" or are "collectively acceptable" [Baroni and Giacomin, 2009]. A crucial assumption of abstract argumentation is that this evaluation of arguments can be done solely on the basis of the abstract representation of the relationship between arguments obtained in the step described in the previous paragraph. The criteria or method used to settle the acceptance of arguments is called a "semantics", by now a plethora of semantics motivated by different theoretical and practical concerns having been developed for the framework for abstract representention defined in [Dung, 1995].

The final step of the framework for formally modelling the process of argumentation ("conflict resolution") can be to derive some conclusion based on the sets of accepted arguments determined by the (alternatively, some) semantics. Continuing with our example, one could reach the (possibly not very useful) conclusion that

$$(\Phi_1 \wedge \alpha_1) \vee (\Phi_2 \wedge \alpha_2) \vee ... \vee (\Phi_n \wedge \alpha_n)$$

where (Φ_i, α_i) for i such that $1 \leq i \leq n$ are all the arguments that are accepted under some semantics. Several reasoning tasks which we will introduce in detail in Section 2.4 can be defined for ADFs and with respect to the different semantics that may aid this last step.

As has already been noted in the introduction to this work, although very general, Dung's AFs do not directly support modelling more complex interactions between arguments as relations of mutual support and sophisticated relations of attack. For this and similar issues, several extensions of AFs have been proposed to date (e.g. those presented in [Bench-Capon, 2003, Cayrol and Lagasquie-Schiex, 2005, Coste-Marquis et al., 2006, Baroni et al., 2011b]), one of the most general being that of abstract dialectical frameworks (ADFs) first proposed in [Brewka and Woltran, 2010].

In [Brewka et al., 2013] ADFs are proposed as a “argumentation middleware”, i.e. a formalism allowing for more straightforward modelling but which can be translated into the framework of argumentation proposed by Dung. In ADFs arguments are replaced with “statements” as the main building block of argumentation frameworks, and acceptance conditions in the form of arbitrary boolean formulas are associated to every statement. The generalisation of the basic semantics given by Dung to the ADF scenario given in [Brewka and Woltran, 2010] has been extended to cover all types of ADFs in [Brewka et al., 2013].

Argumentation frameworks

As indicated previously, an abstract dialectical argumentation framework is a set of statements, together with associated acceptance conditions. Formally:

Definition 2.3.1. An *abstract dialectical framework* (ADF) is a pair $D = (S, C = \{\phi_s\}_{s\in S})$ where S is a set of *statments* and for each $s \in S$, ϕ_s is a propositional formula, the *acceptance condition* associated to s. All the variables occurring in the formulas in $C = \{\phi_s\}_{s\in S}$ represent statements in S. By abuse of notation, we stipulate this to mean that every variable occurring in the acceptance conditions is in S.

Although S in Definition 2.3.1 can in principle be infinite, in this work we will restrict our attention to ADFs where the set of statements is finite.

Example 2.3.1. A very simple example of an ADF is $D = (S, C)$ with $S = \{a, b, c, d\}$ and the acceptance conditions (acceptance conditions in square brackets after the statements they correspond to) [1]:

$$a\,[a],\;\; b\,[b],\;\; c\,[a \wedge \neg b],\;\; d\,[\neg a \wedge b].$$

Acceptance of statements can now be considered analogously to a propositional valuation v but interpreting $v(s) = 1$ to mean “statement s is accepted” and $v(s) = 0$ to mean “statement s is *not* accepted”. With this in mind, the intuition behind Definition 2.3.1 is that “s is accepted

[1] This ADF can be seen as instantiated in a (slightly adapted) fragment of the partially contradicting reports given by various witnesses of the language used by a shrill voice heard at what turned out to be a crime scene in E.A. Poe’s classic locked room mistery “The Murders in the Rue Morgue” [Poe, 2006]. “a” can be seen as denoting the statement given by Henri Duval, a Frenchman, who was certain the shrill voice was not French. “b” then denotes the statment given by Alberto Montani, an Italian, who claims that the shrill voice was definitely not Italian. “c” denotes the statement, again given by Henri Duval (who does not speak Italian), that the shrill voice was, judging from the intonation, probably Italian and “d” the statement by Alberto Montani (who does not speak French) that the words uttered by the shrill voice seemed to him to be French. The ADF given in the example results from, for example, using some modal logic [Blackburn et al., 2001] to represent the different statements, $a := \Box\neg i$, $b := \Box\neg f$, $c := \diamond f$, $d := \diamond i$ (f stands for “French” and i for “Italian”) and considering the attack relation to be: ϕ “attacks” ψ if $\phi \models \neg\psi$, and the “support” relation as: ϕ “supports” ψ if $\phi \models \psi$ (where $\models$ is considered taking in account some background theory from which it follows, in particular, that $\Box\neg i \models \diamond f$ and $\Box\neg f \models \diamond i$) but disallowing self-support of statements, except in those cases in which no other statements support or attack them (in these cases self support indicates that these statements are not a-priori accepted or rejected).

if and only if ϕ_s is satisfied", where ϕ_s makes explicit, in terms of the valuations that satisfy the formula, which (combination of) statements can be accepted and which not in order for s to be accepted.

Abstract dialectical frameworks can also be represented in terms closer to the definition of abstract frameworks introduced by Dung in [Dung, 1995] by making explicit the dependencies between statements which are only implicit in the acceptance conditions associated to each statement in Definition 2.3.1 and by using (boolean) functions instead of propositional formulas for the acceptance conditons. Abstract dialectical frameworks are defined in terms of this "graph based" representation in [Brewka and Woltran, 2010] and [Brewka et al., 2013] (although the previous "propositional" representation is also used there).

Definition 2.3.2. A (*graph based*) abstract dialectical framework is a tuple $D = (S, L, C)$ where

1. S is a set of statements,
2. $L \subseteq S \times S$ is a set of *links*,
3. $C = \{C_s\}_{s \in S}$, the acceptance conditions, is a set of total functions $C_s : 2^{par_L(s)} \rightarrow \{1, 0\}$, one for each statement s. Here $par_L(s)$ are the *parents* of s in L, i.e. $par_L(s) = \{s' \mid (s', s) \in L\}$.

Equivalence betweeen the "graph based" and "propositional" representations can now be defined as follows [Ellmauthaler, 2012]:

Definition 2.3.3. An ADF $D = (S, C)$ is equivalent to a graph based abstract dialectical framework $D' = (S', L, C')$ if and only if $S' = S$, $L = \{(s', s) \mid s', s \in S \text{ and } s' \text{ occurs in } \phi_s\}$ and for each $s \in S$ and $R \subseteq par_L(s)$, $C'_s(R) = 1$ if and only if $v(\phi_s) = 1$ for the valuation v defined as: $v(s') = 1$ if and only if $s' \in R$.

Given an ADF D, this definition gives a way to construct a graph based abstract dialectical framework equivalent to it. Conversely, a graph based ADF $D' = (S', L, C')$ can be translated into a (propositional) ADF $D = (S, C = \{\phi_s\}_{s \in S})$ equivalent to it by setting $S = S'$ and by defining each ϕ_s for each $s \in S$ as follows:

$$\phi_s := \bigvee_{R \subseteq par_L(s) \,:\, C'_s(R)=1} (\bigwedge_{r \in R} r \wedge \bigwedge_{t \in par_L(s) \setminus R} \neg t)$$

Note that while there is only one equivalent graph based ADF for each propositional ADF, there may be more than one propositional ADF equivalent to a given graph based ADF. Therefore, graph based ADFs provide a kind of "standard representation" for ADFs which can be useful for theoretical purposes, although a similar (yet more idiosyncratic) result can be achieved by restricting attention to ADFs with acceptance conditions in certain propositional normal forms, e.g. formulas in CNF.

Example 2.3.2. The graph based ADF corresponding to the ADF in Example 2.3.1 is the ADF $D = (S, L, C)$ with $S = \{a, b, c, d\}$, $L = \{(a, a), (b, b), (a, c), (b, c), (a, d), (b, d)\}$ and $C_a(\emptyset) = 0$, $C_a(\{a\}) = 1$, $C_b(\emptyset) = 0$, $C_b(\{b\}) = 1$, $C_c(\emptyset) = 0$, $C_c(\{a\}) = 1$, $C_c(\{b\}) = 0$, $C_c(\{a, b\}) = 0$, $C_d(\emptyset) = 0$, $C_d(\{a\}) = 0$, $C_d(\{b\}) = 1$, $C_d(\{a, b\}) = 0$.

In this work we will mainly stick to the propositional representation of ADFs since we are especially interested in the connections between ADFs and logic. This representation seems also to fulfill more clearly the role of ADFs as "argumentation middleware" as described in Section 2.3, although a mix of both graph based and propositional representations, as the "hypergraph" representation proposed in [Ellmauthaler, 2012] (especially in graphical form) may be even more useful for modelling purposes.

We now give the definition of argumentation frameworks as defined by Dung in [Dung, 1995] and a translation of "Dung style" argumentation frameworks into ADFs.

Definition 2.3.4. A (*Dung style*) *argumentation framework* (AF) is a pair $F = (A, R)$ where A is a set of *arguments* and $R \subseteq A \times A$.

Some useful concepts for AFs that are also used in the definition of the semantics are the following:

Definition 2.3.5. Let $F = (A, R)$ be an AF. Then

- $a \in A$ *attacks* $b \in A$ *in* F if $(a, b) \in R$.
- $a \in A$ *defends* $b \in A$ from $c \in A$ *in* F if c attacks b and a attacks c.
- $a \in A$ is *defended* by a set $E \subseteq A$ *in* F if for each $b \in A$ that attacks a, there exists some $c \in E$ that defends a from b.
- The *characteristic function* $\mathcal{F}_F$ associated to F is defined as:

  $$\mathcal{F}_F(E) = \{a \in A \mid a \text{ is defended by } E\}$$

  for any $E \subseteq A$.

AFs can be represented as ADFs as follows [Brewka et al., 2013]:

Definition 2.3.6. For an AF $F = (A, R)$, the *ADF associated to F* is the ADF $D_F = (A, C = \{\phi_a\}_{a \in A})$ with $\phi_a = \bigwedge_{b \in par_R(a)} \neg b$ for each $a \in A$.

The "correctness" of this translation can only be stated in terms of the semantics of AFs and ADFs reason for which we defer to Section 2.3 for the statement of this fact which is proven in [Brewka et al., 2013].

Semantics for ADFs: general concept and preliminary notions

A *semantics* (for argumentation) can be described as in [Baroni and Giacomin, 2009] to be "a formal definition of a method (either procedural or declarative) ruling the argument evaluation process" which we have already referred to at the beginning of this section. There are various possible ways to define a semantics, the most popular ones for Dung style argumentation frameworks to date being the so called "extensional" and "labelling-based" approaches.

In the first approach a semantics specifies how to derive from an argumentation framework $F = (A, R)$ a set of extensions $\mathcal{E}$, where an *extension* E is simply a subset of A. Intuitively, each extension gives a set of arguments that can be accepted collectively. In the second approach, a semantics specifies how to derive from an argumentation framework a set of labellings $\mathcal{L}$, where a *labelling* L is a mapping from arguments to a set of predefined labels, usually $\{\text{in}, \text{out}, \text{undecided}\}$. The intuition behind these labels is that the set of arguments labelled "in" under some labelling L are those that can be accepted collectively.

Although the use of more than two labels in the labelling approach in theory allows for finer grained distinctions when definining the semantics, to date equivalent extension-based formulations of labelling-based semantics are usually available [Baroni and Giacomin, 2009] and, hence, the choice of one approach over the other is mainly a matter of convenience. In [Caminada and Gabbay, 2009] the major semantics that have been defined for AFs in the extensional and labelling approach (with three labels) are presented as is the correspondence between both approaches via functions $Ext2Lab$ and $Lab2Ext$ between extensions and labellings that, for most of the semantics, turn out to be inverses of each other.

As has already been noted, ADFs were first introduced in [Brewka and Woltran, 2010] where the standard semantics for Dung's framework were also generalized to this new formalism, although so called "preferred" and "stable" extensions only to the case of a subclass of ADFs called "bipolar". In [Brewka et al., 2013], which is the work we follow in this section to define the semantics of ADFs, the standard semantics for AFs are extended to arbitrary ADFs. This generalization avoids examples [Ellmauthaler, 2012, Strass, 2013] where the stable semantics as defined in [Brewka and Woltran, 2010] leads to unintended results when applied to certain argumentation scenarios.

Semantics for ADFs are defined in [Brewka et al., 2013] essentially generalizing the "labelling based approach" for AFs:

Definition 2.3.7. A *semantics* for an ADF $D = (S, C)$ is a set of *three valued valuations* which are, in turn, mappings from S to $\{1, 0, \frac{1}{2}\}$.

The notion of a "valuation" instead of "labelling" suggests connections between argumentation semantics and logical semantics which are taken advantage of in the definition of the semantics in [Brewka et al., 2013], although the semantics of Kleene's strong three valued logic [Kleene, 1952] is used for this purpose instead of the semantics of classic propositional logic. This is in order to deal with the third value $\frac{1}{2}$. The extension of a three valued valuation to propositional formulas in Kleene's strong three valued logic is defined as follows:

Definition 2.3.8. The value of a propositional formula under a three valued valuation v (in Kleene's strong three valued logic), is a mapping v^* from propositional formulas to $\{1, 0, \frac{1}{2}\}$ defined inductively as follows:

- $v^*(p) = v(p)$ for $p \in \mathcal{P}$.
- $v^*(\top) = 1$ and $v^*(\bot) = 0$.
- For any formula ϕ, $v^*(\neg\phi) = 1$ if $v^*(\phi) = 0$, $v^*(\neg\phi) = 1$ if $v^*(\phi) = 0$, and $v^*(\neg\phi) = \frac{1}{2}$ otherwise.
- For any formulas ϕ and ψ, $v^*(\phi \wedge \psi) = 1$ if $v^*(\phi) = 1$ and $v^*(\psi) = 1$, $v^*(\phi \wedge \psi) = 0$ if either $v^*(\phi) = 0$ or $v^*(\psi) = 0$, and $v^*(\phi \wedge \psi) = \frac{1}{2}$ otherwise.

- For any formulas ϕ and ψ, $v^*(\phi \vee \psi) = 1$ if either $v^*(\phi) = 1$ or $v^*(\psi) = 1$, $v^*(\phi \vee \psi) = 0$ if $v^*(\phi) = 0$ and $v^*(\psi) = 0$, and $v^*(\phi \vee \psi) = \frac{1}{2}$ otherwise.

In the context of the semantics for ADFs, acceptance conditions in ADFs involving defined connectives are evaluated by first translating these into formulas involving only $\neg$, $\wedge$, and $\vee$ using the (classical) definitions of the connectives given in Section 2.1. As for classical propositonal logic, we will ommit the $*$ when referring to an extension of a three valued valuation to an arbitrary propositional formula in the rest of this work.

In order to define the semantics for ADFs, the values 1, 0, $\frac{1}{2}$ are ordered by $\leq_i$ according to their "information content":

Definition 2.3.9. $x <_i y$ ("y has *greater information content* than x") if $(x, y) \in <_i$ where $<_i := \{(\frac{1}{2}, 1), (\frac{1}{2}, 0)\}$. $x \leq_i y$ ("y has *greater or equal information content* than x") if $x <_i y$ or $x = y$. $x <>_i y$ ("x and y are *incomparable with respect to their information content*") if neither $x <_i y$, $y <_i x$ nor $x = y$ (i.e. $x = 1$ and $y = 0$ or $x = 0$ and $y = 1$).

The pair $(\{1, 0, \frac{1}{2}\}, \leq_i)$ forms a *complete meet-semilatice*, i.e. $\leq_i$ is reflexive, antisymmetric, and transitive on $\{1, 0, \frac{1}{2}\}$ and every non-empty finite set $B \subseteq \{1, 0, \frac{1}{2}\}$ has a greatest lower bound and every nonempty directed set $C \subseteq \{1, 0, \frac{1}{2}\}$ has a least upper bound. A subset is *directed* if any two if its elements have an upper bound in the set.

We remind the reader that a *lower bound* in a partially ordered set Z for a set $B \subseteq Z$ is an element of Z that is less or equal than any of the elements in B, while an *upper bound* is greater or equal than any of the elements in B. The *greatest lower bound* or *meet* is the greatest of all lower bounds of B and is denoted $\sqcap B$, while the *least upper bound* of a set B is the least of all upper bounds of B and is denoted $\sqcup B$. The meet for $(\{1, 0, \frac{1}{2}\}, \leq_i)$ can be read as "consensus" and assigns $1 \sqcap 1 = 1$, $0 \sqcap 0 = 0$ and returns $\frac{1}{2}$ otherwise.

The greatest lower bound of a set B is the *least element* if it is in B, while the *greatest element* of a set B is a least upper bound that is in B. We denote the greatest element by g.e. B and the least element by l.e. B. Finally, an element is *minimal* for a set Z if there is no element in Z that is less than it, while an element is *maximal* if there is no element in Z that is greater than it.

In order to define the semantics for ADFs, the information ordering $\leq_i$ is extended to valuations v_1, v_2 over a set of statements S by defining:

$$v_1 \leq_i v_2 \text{ if } v_1(s) \leq_i v_2(s)$$

for all $s \in S$. The set of all three valued valuations over S also forms a complete meet-semilattice with respect to the information ordering $\leq_i$. The consensus meet operation $\sqcap$ of this semilattice is given by:

$$(v_1 \sqcap v_2)(s) = v_1(s) \sqcap v_2(s)$$

for all $s \in S$. The least element of this semilattice is the valuation mapping all statements to $\frac{1}{2}$, the *two valued valuations* being the $\leq_i$ maximal elements. The latter are those valuations v for which it holds that for no $s \in S$, $v(s) = \frac{1}{2}$.

A two valued valuation w can be said to *extend* a three valued valuation v if and only if $v \leq_i w$. The set of two valued valuations that extend a three valued valuation v can then be denoted as $[v]_2$. The elements of $[v]_2$ form a $\leq_i$ *antichain* (a subset of a partially ordered set such that any two elements of the subset are incomparable) with greatest lower bound $v = \sqcap [v]_2$.

Finally, in the definitions of most of the semantics for ADFs as we present them in the following section the operator over three valued interpretations we define next plays an important role:

Definition 2.3.10. Given an ADF $D = (S, C = \{\phi_s\}_{s \in S})$, the *characteristic* operator Γ_D *of* D over three valued valuations on S returns, for each three valued valuation v on S, a three valued valuation $\Gamma_D(v)$ defined as follows:

$$\Gamma_D(v)(s) := \sqcap \{w(\phi_s) \mid w \in [v]_2\}$$

for each $s \in S$.

For each statement $s \in S$, $\Gamma_D(v)$ returns the "consensus" value for the statment's acceptance formula ϕ_s, where the "consensus" value is established by taking the meet over the evaluation of ϕ_s by each two valued valuation that extends v.

Definition of major semantics for ADFs

We now turn to the definition of some of the most important semantics that have been defined for ADFs. As indicated before, we here follow the definitions of the semantics as given in [Brewka et al., 2013]. This is mainly because we are interested in declarative definitions which can be stated more directly in formal logic, but see [Strass, 2013] for a more recent formulation of the semantics of ADFs by associating with ADFs "characteristic one-step consequence operators" and defining the various semantics as different fixpoints of these.

A basic requirement for acceptance of statements seems to be that whenever a statement's acceptance status is established (i.e. a statement is mapped to 1 or 0), its acceptance status be "consistent with the acceptance conditions". This is captured formally by the notion of a "three valued model":

Definition 2.3.11. A *three valued model* of an ADF $D = (S, C = \{\phi_s\}_{s \in S})$ is a three valued valuation such that for all $s \in S$, if $v(s) \neq \frac{1}{2}$, then $v(s) = v(\phi_s)$.

To give examples of the application of the semantics to ADFs, given an order $\{s_1, s_2, ..., s_n\}$ of the statements of an ADF, we use the notation $\{v_{s_1}, v_{s_2}, .., v_{s_n}\}$ to present valuations. Here each $v_{s_i} \in \{1, 0, \frac{1}{2}\}$ $(1 \leq i \leq n)$ presents the assignment given to s_i by the valuation in question.

Example 2.3.3. The three valued models of the ADF of Example 2.3.1 are: $\{1,1,0,0\}$, $\{1,1,\frac{1}{2},0\}$, $\{1,1,0,\frac{1}{2}\}$, $\{1,1,\frac{1}{2},\frac{1}{2}\}$, $\{0,1,0,1\}$, $\{0,1,\frac{1}{2},1\}$, $\{0,1,0,\frac{1}{2}\}$, $\{0,1,\frac{1}{2},\frac{1}{2}\}$, $\{\frac{1}{2},1,0,\frac{1}{2}\}$, $\{\frac{1}{2},1,\frac{1}{2},\frac{1}{2}\}$, $\{1,0,1,0\}$, $\{1,0,1,\frac{1}{2}\}$, $\{1,0,\frac{1}{2},0\}$, $\{1,0,\frac{1}{2},\frac{1}{2}\}$, $\{1,\frac{1}{2},\frac{1}{2},0\}$, $\{1,\frac{1}{2},\frac{1}{2},\frac{1}{2}\}$, $\{\frac{1}{2},0,\frac{1}{2},0\}$, $\{\frac{1}{2},0,\frac{1}{2},\frac{1}{2}\}$, $\{0,0,0,0\}$, $\{0,0,\frac{1}{2},0\}$, $\{0,0,0,\frac{1}{2}\}$, $\{0,0,\frac{1}{2},\frac{1}{2}\}$, $\{\frac{1}{2},\frac{1}{2},\frac{1}{2},\frac{1}{2}\}$, $\{0,\frac{1}{2},\frac{1}{2},\frac{1}{2}\}$, $\{0,\frac{1}{2},0,\frac{1}{2}\}$.

We now introduce semantics for ADFs which are known to generalise some of the most important semantics that have been defined for AFs. By "generalise" we mean the following:

Definition 2.3.12. Let F be an AF and D_F its associated ADF.

- For a valuation v, the set $E_v := \{s \in S | v(s) = 1\}$ defines the unique *extension associated with* v.
- The notion of a σ-valuation for ADFs *generalises* that of a ς-extension for AFs, if for every ADF D and valuation v, v is a σ-valuation for D if and only if E_v is a ς-extension for D_F.

The notion of an "admissible valuation", generalises in the ADF scenario, the concept of an "admissible extension" of an AF. The intuition behind this last notion is that an admissible extension can "stand together" and "on its own", i.e. no two arguments in the extension should attack each other and the set should be able to withstand attacks received from other arguments by "replying" with other attacks [Baroni and Giacomin, 2009]. Formally, the definition of an "admissible set" for an AF, which in turn depends on the central notion of a "conflict free set", is as follows:

Definition 2.3.13. Let $F = (A, R)$ be an AF.

- A *conflict free set* for F is a set $E \subseteq A$ such that for no two a, b in E, a *attacks* b.
- An *admissible extension* for F is a conflict free set $E \subseteq A$ such that each $a \in E$ is defended by E.
- An equivalent definition of an admissible extension is that it is a set $E \subseteq A$ that is conflict free and such that $E \subseteq \mathcal{F}_F(E)$.

The definition of an "admissible valuation" for an ADF now is:

Definition 2.3.14. A three valued valuation v for an ADF $D = (S, C)$ is *admissible in* D if $v \leq_i \Gamma_D(v)$.

More or less intuitively, this means that an admissible valuation v assigns a value from $\{1, 0, \frac{1}{2}\}$ to every statement $s \in S$ that is "at most as much commited" with respect to the order $\leq_i$ as the "consensus" among all evaluations of the acceptance condition associated to s by all two valued valuations that extend v.

Example 2.3.4. For the ADF of Example 2.3.1 the admissible valuations coincide with the three valued models (Example 2.3.3).

The notion of a "complete valuation", generalises for ADFs, the concept of a "complete extension" for an AF. The intuition behind this last notion is that a complete extension is a set of arguments "which is able to defend itself and includes all arguments it defends" [Baroni and Giacomin, 2009]. Formally:

Definition 2.3.15. Let $F = (A, R)$ be an AF.

- A *complete extension* for F is a set $E \subseteq A$ that is admissible for F and includes every $a \in A$ that is defended by E.
- An equivalent definition of a complete extension is that it is a set $E \subseteq A$ that is conflict free and such that $E = \mathcal{F}_F(E)$.

Mirroring the structure of the latter formulation of the definition of complete extensions for AFs, the definition of a "complete valuation" for an ADF is:

Definition 2.3.16. A three valued valuation v for an ADF $D = (S, C)$ is *complete in* D if $v = \Gamma_D(v)$.

Example 2.3.5. The complete valuations of the ADF of Example 2.3.1 are: $\{1,1,0,0\}$, $\{0,1,0,1\}$, $\{\frac{1}{2},1,0,\frac{1}{2}\}$,$\{1,0,1,0\}$,$\{1,\frac{1}{2},\frac{1}{2},0\}$,$\{\frac{1}{2},0,\frac{1}{2},0\}$,$\{0,0,0,0\}$,$\{\frac{1}{2},\frac{1}{2},\frac{1}{2},\frac{1}{2}\}$,$\{0,\frac{1}{2},0,\frac{1}{2}\}$.

As is to be expected by now, the notion of a "preferred valuation", generalises for ADFs, the concept of a "preferred extension" for an AF. The intuition behind this last notion is that a preferred extension is a set of arguments "which is as large as possible and able to defend itself from attacks" [Baroni and Giacomin, 2009]. Formally:

Definition 2.3.17. Let $F = (A, R)$ be an AF.

- A *preferred extension* for F is a set $E \subseteq A$ that is admissible for F and such that if $T \subseteq A$ is admissible for F, then $E \not\subset T$.
- An equivalent definition of a preferred extension is that it is a set $E \subseteq A$ that is maximal admissible for F.

The definition of a "preferred valuation" for an ADF is:

Definition 2.3.18. A three valued valuation v for an ADF $D = (S, C)$ is *preferred in* D if it is $\leq_i$-maximal admissible.

Example 2.3.6. The preferred valuations of the ADF of Example 2.3.1 are: $\{1,1,0,0\}$, $\{0,1,0,1\}$, $\{1,0,1,0\}$, $\{0,0,0,0\}$.

Two valued models for ADFs generalise stable extensions for AFs which can be understood intuitively to be a set of arguments "that attacks all arguments not included in it" [Baroni and Giacomin, 2009] and are particulary notable because they can be put in correspondence with solutions of cooperative n-person games, the stable marriage problem, extensions of Reiter's default logic [Reiter, 1980], as well as stable models of logic programs [Gelfond and Lifschitz, 1988]. The formal definition is:

Definition 2.3.19. Let $F = (A, R)$ be an AF. A *stable extension* for F is a set $E \subseteq A$ that is conflict free for F and such that for each $b \in A \setminus E$ there exists a $a \in E$ that attacks b.

The definition of a "two valued model" of an ADF is essentially that of a three valued model where no statement is mapped to $\frac{1}{2}$:

Definition 2.3.20. A three valued valuation v for an ADF $D = (S, C = \{\phi_s\}_{s \in S})$ is a *two valued model of* D if for every $s \in S$, $v(s) = v(\phi_s)$.

Example 2.3.7. The two valued models of the ADF of Example 2.3.1 coincide with the preferred valuations (Example 2.3.6).

Although two valued models for ADFs in effect generalise stable extensions for AFs, it should be noted that a new generalisation of this notion has been proposed in [Brewka et al., 2013] in the light of anomalies that have been found when attempting to apply the notion of two valued model to some examples. These anomalies run counter-intuitive to what is to be expected from a "stable" valuation given known properties of "stable" extensions of AFs and, hence, it is also only the new generalisation of stable extensions to ADFs that is called "stable".

Finally, grounded valuations for ADFs generalise grounded extensions for AFs. Underlying the latter is a "procedural" intuition; given an AF $F = (A, R)$ and starting with $Z = \emptyset$, the grounded extension results from following the procedure:

1. put each argument $a \in A$ which is not attacked in F into Z; if no such argument exists, return S;
2. remove from F all (new) arguments in Z and all arguments attacked by them and continue with step 1.

This turns out to be equivalent to the following definition:

Definition 2.3.21. Let $F = (A, R)$ be an AF. The *grounded extension* for F is the least fixpoint of $\mathcal{F}_F$.

We remind the reader that a fixpoint of a function F is an argument x such that $F(x) = x$. Mirroring the previous definition, a three valued valuation is grounded for an ADF D if it is the least fixpoint of Γ_D and it is guaranteed to exist because Γ_D is $\leq_i$-monotone [Brewka et al., 2013]:

Definition 2.3.22. The *grounded* valuation v for an ADF $D = (S, C = \{\phi_s\}_{s \in S})$ is the least fixpoint of Γ_D.

Example 2.3.8. The grounded valuation of the ADF of Example 2.3.1 is $\{\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}\}$. Note that, if the acceptance condition of statement a is set to $\top$, the grounded valuation is $\{1, \frac{1}{2}, \frac{1}{2}, 0\}$, while if both the acceptance conditions of a and b are set to $\top$ the grounded valuation is $\{1, 1, 0, 0\}$ [2].

The following theorem (from [Brewka et al., 2013]) summarises some of the relations that exists between the semantics we have defined for ADFs and some of the most important semantical notions for AFs:

[2]The latter corresponds to the conclusion arrived at by the aficionado detective Auguste Dupin in Poe's story (see Footnote 1) , who assumes the witnesses are credible when it comes to identifying their own mother tongue and is able to deduce (using also other evidence in the story) that the source of the shrill voice and also the "murderer" at the crime scene in the story is a fugitive gorilla.

Theorem 2.3.1. Two valued models and admissible, complete, preferred, grounded, valuations for ADFs generalise stable, admissible, complete, preferred, grounded extensions respectively for AFs.

Theorem 2.3.1, via the notion of "generalisation" given in Definition 2.3.12, justifies the notion of an ADF being "associated" to an AF as defined in Definition 2.3.6.

The following theorem [Brewka et al., 2013] summarises the relations that exist between the different semantics for ADFs, which in fact also reflects the relationship that exist between the semantics of AFs that the different semantics for ADFs generalise.

Theorem 2.3.2. Let D be an ADF. The following inclusions hold:

$$2mod(D) \subseteq pref(D) \subseteq comp(D) \subseteq adm(D)$$

where $2mod(D)$,$pref(D)$, $comp(D)$, $adm(D)$ denote the sets of two-valued models, and preferred, complete, admissible valuations respectively.

It should finally also be clear from the definition of grounded and complete valuations that the following theorem [Brewka et al., 2013] holds:

Theorem 2.3.3. Let D be an ADF. The grounded valuation for D is the $\leq_i$-least complete valuation for D.

2.4 Complexity

In this section we first of all define the *decision problems* associated to ADFs on the one hand and QBFs on the other hand that play an important role in the present work. Then we review the complexity of these reasoning tasks, not without first giving a very brief overview of complexity theory.

The material in this section plays a central role in our work for several reasons. In the first place, a central concern of the present work is in *encoding* reasoning problems associated to ADFs into reasoning tasks defined in quantified boolean logic and the notion of an "encoding" is essentially that of a "reduction" which we introduce here. In the second place, we are interested in giving encodings of the decision problems defined for ADFs that are "at the right level" of complexity and the material covered in this section provides the necessary theoretical framework for doing so. Finally, this section also makes more precise the relationship between propositional logic and quantified boolean logic that we have hinted at in previous sections.

The focus of Section 2.3 has been what can be called the problem of "evaluating" ADFs with respect to different semantics or the problem of enumerating the valuations of a given type for some ADF. An analogous problem exists for quantified boolean formulas: to enumerate the models of a QBF.

Among other important reasoning tasks that can be defined for ADFs are many *decision problems*. These can be characterised informally as a set of problem instances and a yes-no question regarding these problem instances. Some of the central decision problems that have been defined for ADFs are the following:

- EXISTS$_\sigma$: Given an ADF D and a valuation type σ, does there exist a σ valuation for D?
- EXISTS$_\sigma^{\neg\emptyset}$: Given an ADF $D = (S, C)$ and a valuation type σ, does there exist a σ valuation v for D such that for some $s \in S$, $v(s) \neq \frac{1}{2}$?
- CRED$_\sigma$: Given an ADF $D = (S, C)$, a valuation type σ, and a statement $s_* \in S$, does there exist a σ valuation for D such that $v(s_*) = 1$?
- SKEPT$_\sigma$: Given an ADF $D = (S, C)$, a valuation type σ, and a statement $s_* \in S$, is it the case that for all σ valuations for D, $v(s_*) = 1$?

Analogous decision problems can clearly also be defined for AFs. A central decision problem for quantified boolean formulas is determining the satisfiability of a QBF:

- QSAT: Given a QBF ϕ, is ϕ satisfiable?

SAT is the same problem but restricting attention to propositional formulas.

The traditional concern of complexity theory has been to study decision problems from the perspective of the resources (running time, memory space) required to solve them. One of the main objectives has, in particular, been to classify computational problems from this perspective. More specifically, *structural complexity theory* which we consider here analyses the difficulty of problems in terms of the worst-case behavior (in terms of running time, memory space) of algorithms that solve them.

Worst case behaviour is measured as a function of the input-size. Also, in order to abstract from hardware specifics, algorithms are specified for Turing Machines, a prominent model of computation introduced by Turing in [Turing, 1936] to formalise the notion of an “algorithm”. An important distinction for complexity theory concerns *deterministic* and *non-deterministic* Turing machines. Though equally powerful from the perspective of the algorithms they can implement, these are relevant when considering the complexity of algorithms since to date no efficient (polynomial) simulation of non-determinism in a deterministic Turing machine has been shown to exist and it is widely assumed that it in fact does not.

The central theoretical tool in complexity theory is the notion of *reduction*. A reduction from a problem P_1 to a problem P_2 is a function f from the instances of problem P_1 to the instances of problem P_2 which is efficiently computable (for the purposes of this work: requires polynomial time) and which satisfies the following property: an instance i of P_1 has answer “yes” with respect ot the problem P_1 if and only if the intance $f(i)$ of P_2 has answer “yes” with respect to the problem P_2.

The notion of reduction allows to order decision problems as follows: $P_1 \leq_R P_2$ if and only if there exists a reduction from P_1 to P_2. The intuition behind this ordering is that if there exists a reduction f from P_1 to P_2 then P_1 is not more difficult than P_2, since one can solve an instance of i of P_1 by solving $f(i)$ in P_2. The latter means that in case well developed tools are available for solving P_2, a reduction can have significant practical consequences as well as being of theoretical interest.

A problem P is now called *hard* for a complexity class C if every problem in C can be reduced to P. It is *complete* for C (this can be written: C-c) if it also belongs to C. This notion makes more precise the notion of a problem being "prototypical" for a complexity class that we have already used in this work. The complement of a class C is denoted co-C.

Some important complexity classes relevant for the present work are in the first place the class L of problems that can be answered by a deterministic Turing Machine using logarithmic space. P, on the other hand, is the class of problems that can be answered by a deterministic Turing machine in polynomial time. It holds that L $\subseteq$ P. The same definition for P but for non deterministic Turing machines leads to the class NP. As has been hinted at in a previous paragraph, it is generally assumed that P $\subset$ NP. While problems in class P are *tractable*, problems in NP are assumed to be *intractable*, i.e. they require exponential time in the worst case. In [Cook, 1971] it is shown that SAT is NP-complete.

The class PSPACE is the set of problems that can be answered by a deterministic Turing machine using polynomial *space*. It is an open question but usually assumed that NP $\subset$ PSPACE. In [Stockmeyer and Meyer, 1973] it is shown that QSAT is PSPACE-complete.

The *polynomial hierarchy* is a family of complexity classes within PSPACE introduced in [Stockmeyer and Meyer, 1973, Stockmeyer, 1976]. It is defined as follows for $k \geq 0$:

$$\Delta_0^P := \Sigma_0^P := \Pi_0^P := P$$

$$\Sigma_{k+1}^P := NP^{\Sigma_k^P}, \Pi_{k+1}^P := co{-}\Sigma_{k+1}^P, \Delta_{k+1}^P := P^{\Sigma_k^P}$$

Δ_{k+1}^P (respectively, Σ_{k+1}^P) is the class of all problems that can be answered deterministically (respectively, non deterministically) in polynomial time with the help of an oracle for a problem in Σ_k^P. An oracle is a subroutine which solves a problem in the complexity class Σ_k^P in constant time. In particular, one has that NP $= \Sigma_1^P$, co-NP $= \Pi_1^P$ and P $= \Delta_1^P$. The polynomial hierarchy PH is defined as the union $\cup_{k=0}^{\infty}\Sigma_k^P$.

The polynomial hierarchy is relevant for the present work because for each $k \geq 1$, prototypical problems for Σ_k^P and Π_k^P can be found by specialising the QSAT problem to particular kinds of QBFs in prenex normal form according to their prefix. We first give the definition which allows for classification of QBFs in prenex normal form according to their prefix:

Definition 2.4.1. (*Prefix type of a QBF*) Every propositional formula has the prefix type $\Sigma_0 = \Pi_0$. Let ϕ be a QBF with prefix type Σ_n (respectively, Π_n), then the formula $\forall x_1 \ldots \forall x_m \phi$ (respectively $\exists y_1 \ldots \exists y_m \phi$) is of type Π_{n+1} (respectively Σ_{n+1}) for any $m > 0$.

The result hinted at in the previous paragraph can now be stated formally:

Theorem 2.4.1. For $k \geq 1$, the satisfiability problem for QBFs with prefix type Σ_k is Σ_k^P-complete, and for formulas with prefix type Π_k, it is Π_k^P-complete.

Tables 2.1 and 2.2 summarise the complexity of the decision problems for the semantics that are of interest for the present work. Sources for the results regarding AFs are [Dung, 1995, Dimopoulos and Torres, 1996, Dunne and Bench-Capon, 2002, Coste-Marquis et al., 2005, Dvořák and Woltran, 2011]. Almost all of the results we refer to for ADFs are of a very recent date [Strass and Wallner, 2013, Strass and Wallner, 2014], except for NP-completeness of existence (and non-trivial existence) of two valued models which is proven in [Brewka et al., 2013] and the non-trivial complexity results regarding three valued models which we prove in Section 3.2. Note that the complexity of many of the decision problems for ADFs "jump" one level of the polynomial hierarchy with respect to the problems for AFs they generalise, with skeptical acceptance for preferred semantics, in particular, being Π_3^P-complete.

	admissible	complete	preferred	grounded	stable
EXISTS$_\sigma$	trivial	trivial	trivial	trivial	NP-c
EXISTS$_\sigma^{\neg\emptyset}$	NP-c	NP-c	NP-c	in L	NP-c
CRED$_\sigma$	NP-c	NP-c	NP-c	P-c	NP-c
SKEPT$_\sigma$	trivial	P-c	Π_2^P-c	P-c	co-NP-c

Table 2.1: Summary of complexity of reasoning for AFs

	3-model	admissible	complete	preferred	grounded	2-model
EXISTS$_\sigma$	trivial	trivial	trivial	trivial	trivial	NP-c
EXISTS$_\sigma^{\neg\emptyset}$	NP-c	Σ_2^P-c	Σ_2^P-c	Σ_2^P-c	co-NP-c	NP-c
CRED$_\sigma$	NP-c	Σ_2^P-c	Σ_2^P-c	Σ_2^P-c	co-NP-c	NP-c
SKEPT$_\sigma$	trivial	trivial	co-NP-c	Π_3^P-c	co-NP-c	co-NP-c

Table 2.2: Summary of complexity of reasoning for ADFs

We have already indicated where the proofs of the results contained in Tables 2.1 and 2.2 can be found, but as conclusion to this chapter a brief (and informal) elucidation about some of the more "easy" complexity results regarding ADFs that are relevant for the present work follow. In these comments we abbreviate "3-model" as "3mod", "2-model" as "2-mod", "admissible" as "adm", "complete" as "comp", "preferred" as "pref", and "grounded" as "ground".

- EXISTS$_{3mod}$, SKEPT$_{3mod}$, EXISTS$_{adm}$ and SKEPT$_{adm}$ are trivial because the valuation mapping all statements to $\frac{1}{2}$ is a three valued model as well as admissible for any ADF.

- As indicated previously, we prove the NP-completeness of EXISTS$_{3mod}^{\neg\emptyset}$ and CRED$_{3mod}$ by reduction from EXISTS$_{2mod}$ and CRED$_{2mod}$ respectively in Section 3.2.

- EXISTS$_{2mod}$ is NP-complete because there is no distinction between trivial and non-trivial valuations for two-valued models.

- EXISTS$_{ground}$ is trivial because the least fixpoint of the characteristic operator is guaranteed to exist for any ADF.

- CRED$_{ground}$ and SKEPT$_{ground}$ have the same complexity because they are equivalent, given the uniqueness of the grounded valuation for any ADF.

- EXISTS$_{comp}$ and EXISTS$_{pref}$ are trivial because of the existence of the grounded valuation (which is the $\leq_i$-least complete valuation; see Theorem 2.3.3) and an admissible valuation (from which, given that the set of admissible valuations is finite, the existence of a maximal admissible valuation follows) for any ADF respectively.

- SKEPT$_{comp}$ is co-NP-complete because it is equivalent to SKEPT$_{ground}$ given that, since the grounded valuation $ground$ is the $\leq_i$-complete valuation for any ADF $D = (S, C)$, it is sufficient to determine that $ground(s_*) = 1$ in order to conclude that all complete valuations assign 1 to $s_* \in S$.

- EXISTS$^{\neg\emptyset}_{comp}$, CRED$_{comp}$, EXISTS$^{\neg\emptyset}_{pref}$, CRED$_{pref}$ are Σ_2^P-c because these decision problems are equivalent to the corresponding problems for admissible valuations. The reason for this is that any admissible valuation v for which $v(s_*) = 1$ or $v(s_*) = 0$ for some $s_* \in S$ (for an ADF $D = (S, C)$) can be extended to a complete (given its monotonicity, via repeated application of the fixpoint operator) or a maximal admissible valuation (given that the set of admissible valuations is finite) for which the same holds.

CHAPTER 3

Encodings

In this chapter we present polynomial-time reductions or encodings of some of the main reasoning tasks associated to ADFs into some of the main reasoning tasks associated to QBFs. On the one hand, these encodings enable to solve the problems related to ADFs via solutions to the corresponding problems in the QBF scenario. On the other hand, these encodings provide a *uniform axiomatization* of all the reasoning problems associated to ADFs we consider here into the relatively simple and well understood setting of quantified boolean logic.

Specifically, in the first place we present encodings of the valuation enumeration problem for ADFs into the model enumeration problem for QBFs. We recall the definition of both problems here:

- ENUM$_\sigma$: Given an ADF D, and a valuation type σ, which are all the σ valuations for D?
- QENUM: Given a QBF ϕ, which are all the models of ϕ?

In the second place, we present encodings of various of the main decision problems defined for ADFs into QSAT. Specifically, we present encodings for the existence, non-trivial existence, credulous acceptance, and skeptical acceptance problems for three valued ("3mod") and two valued ("2mod") models, as well as admissible ("adm"), complete ("comp"), preferred ("pref"), and grounded ("ground") valuations for ADFs. We recall the definition of these problems here:

- EXISTS$_\sigma$: Given an ADF D and a valuation type σ, does there exist a σ valuation for D?
- EXISTS$_\sigma^{\neg\emptyset}$: Given an ADF $D = (S, C)$ and a valuation type σ, does there exist a σ valuation v for D such that for some $s \in S$, $v(s) \neq \frac{1}{2}$?
- CRED$_\sigma$: Given an ADF $D = (S, C)$, a valuation type σ, and a statement $s_* \in S$, does there exist a σ valuation for D such that $v(s_*) = 1$?
- SKEPT$_\sigma$: Given an ADF $D = (S, C)$, a valuation type σ, and a statement $s_* \in S$, is it the case that for all σ valuations for D, $v(s_*) = 1$?

We finally also recall the definition of QSAT:

- QSAT: Given a QBF ϕ, is ϕ satisfiable?

In the most general terms, and somewhat informally, an *encoding* from a problem P_1 to a problem P_2 is simply a function from the instances of problem P_1 to the instances of problem P_2, the notion of "reduction" we introduced in Section 2.4 being a special case of this concept. Some of the main properties an encoding from a decision problem to QSAT can have and that we are interested in this work are the following:

Definition 3.0.2. An encoding $\mathcal{E}$ from a problem P_1 to QSAT is

1. *polynomial* if for each instance i of P_1, $\mathcal{E}(i)$ is computable in polynomial time in the size of i.
2. *faithful* if for each instance i of P_1, $\mathcal{E}(i)$ is satisfiable if and only if i is a yes-instance of P_1.
3. *indicates complexity* if for each instance i of P_1, $\mathcal{E}(i)$ is a QBF of type Σ_n or Π_n for some $n \geq 0$.
4. *reflects complexity* if it indicates complexity and QSAT for the prefix type of $\mathcal{E}(i)$ for each instance i of P_1, as well as P_1 are C-complete for the same complexity class C.
5. *adequate* if it satisfies 1,2,3, and 4.

All the encodings of decision problems regarding ADFs to QSAT we present in this work are adequate. It should be noted that to show that an encoding indicates complexity can also be carried out in two stages, i.e. by showing that the encoding in question yields a QBF which, in turn, can be translated in polynomial time into a QBF in prenex normal form.

In the case of the encodings from the valuation enumeration problems for ADFs to the model enumeration problem for QBFs, in this work we only require that the encodings be polynomial in the same sense as in Definition 3.0.2 and that they be faithful in the following manner:

Definition 3.0.3. Given a QBF ϕ, let $\mathcal{Z}|_{\text{FREE}(\phi)} := \{z|_{\text{FREE}(\phi)} \mid z \text{ is a valuation (on QBFs)}\}$. An encoding $\mathcal{E}$ from ENUM$_\sigma$ to QENUM is *faithful* if for each instance D of ENUM$_\sigma$ there is a one to one correspondence f between $\mathcal{Z}|_{\text{FREE}(\mathcal{E}(D))}$ and the valuations on D such that z is a model of $\mathcal{E}(D)$ if and only if $f(z|_{\text{FREE}(\mathcal{E}(D))})$ is a σ valuation for D.

For the different types of valuations we consider, most of the encodings we define in this work are based on an encoding that, given an ADF D, gives an open QBF Φ^σ_D such that there is a one to one correspondence between the σ valuations of D and the (two valued) models of Φ^σ_D, modulo the assignments given to the variables that are not free or do not occur in Φ^σ_D. For each type of valuation σ, we call this formula Φ^σ_D the *defining formula for σ valuations*. In particular, the defininig formula for a given type of valuation σ allows us to encode the enumeration problem for σ in a straightforward manner.

The semantics for ADFs refer to relatively complex conditions regarding valuations for ADFs and are defined in terms of Kleene's strong three valued logic. To be able to express statements about the semantics as QBFs therefore requires some preliminary work which we carry out in the next section which serves as technical underpinning of the remaining sections of this chapter which present the encodings.

Regarding the presentation of the encodings themselves, in each of the sections that present the various encodings (Sections 3.2 to 3.6), we first give the defining formula for the semantics in question which allows us to provide the encodings for the enumeration problem as indicated previously and then proceed to define the encodings of the different decision tasks we consider in this work (which will often be based on the defining formula). To improve readibility in all the proofs of the following sections we often make use of basic and relatively intuitive properties of propositional and quantified boolean logic stated in Sections 2.1 and 2.2 without mentioning them explicitely.

3.1 Encoding statements about three valued valuations for ADFs as QBFs

As has already been indicated, most of the semantics defined for ADFs make reference to relatively complex conditions referring to valuations for ADFs. To be able to express statements about valuations for ADFs as QBFs we first of all use "signed" variables from as many disjoint sets of variables $S_1^{\pm}$,$S_2^{\pm}$,..., $S_n^{\pm}$ as valuations we need to refer to, where each such set $S_j^{\pm}$ for $1 \leq j \leq n$ is defined as follows

$$S_j^{\pm} := \{s_j^{\oplus} \mid s \in S\} \cup \{s_j^{\ominus} \mid s \in S\}$$

Intuitively, here $s_j^{\oplus}$ stands for "s is accepted" and $s_j^{\ominus}$ for "s is rejected" under some valuation v_j.

Secondly, since the semantics for ADFS make use of Kleene's three valued propositional logic (see Section 2.3) in our encodings we also need to be able to express statements about this logic as QBFs. There are various ways this can been achieved (see, for example, [Arieli and Denecker, 2003, Besnard et al., 2005, Arieli, 2007]); we here mainly adapt the procedure used for encoding statements about three valued labellings on AFs used in [Arieli and Caminada, 2012, Arieli and Caminada, 2013] to our setting.

To distinguish three valued valuation for ADFs from two valued valuations for QBFs, from now on in this chapter we refer to three valued valuations with a superscript "three" and two valued valuations with a superscript "two". There exists a one to one correspondence between three valued valuations on a set of statments of an ADF S and *coherent* two valued valuations *on* on any set of signed variables $S_j^{\pm}$, modulo the the assignments given to variables not in $S_j^{\pm}$, where a coherent two valued valuation on $S_j^{\pm}$ is defined as follows:

Definition 3.1.1. A two valued valuation z^2 is *coherent on a set of (signed) variables* $S_j^{\pm}$ $(j \geq 1)$ if there is no $s \in S$ such that $z^2(s_j^{\oplus}) = 1$ and $z^2(s_j^{\ominus}) = 1$. z^2 is *inconsistent on* $S_j^{\pm}$ if it is not *coherent* on $S_j^{\pm}$.

The mentioned correspondence is now given by the following notion:

Definition 3.1.2. A valuation z^2 which is coherent on a set of variables $S_j^{\pm}$ $(j \geq 1)$ is *induced by* (or *associated with*) a valuation v^3 on S *via the variables* $S_j^{\pm}$ and a valuation v^3 is, in turn, *induced by* (or *associated with*) z^2 *via the variables* $S_j^{\pm}$ if the following conditions hold for every $s \in S$:

a) $v^3(s) = 1$ if and only if $z^2(s_j^{\oplus}) = 1$ and $z^2(s_j^{\ominus}) = 0$,

b) $v^3(s) = 0$ if and only if $z^2(s_j^{\oplus}) = 0$ and $z^2(s_j^{\ominus}) = 1$, and

c) $v^3(s) = \frac{1}{2}$ if and only if $z^2(s_j^{\oplus}) = 0$ and $z^2(s_j^{\ominus}) = 0$.

That z^2 and v^3 are associated via $S_j^{\pm}$ is written as $z^2 \;{}_{S_j^{\pm}}\!\rightleftarrows_S\; v^3$.

Note that since in Definition 3.1.2 nothing is indicated about the assignments of z^2 to the variables not in $S_j^{\pm}$ there are many coherent valuations on $S_j^{\pm}$ induced by a valuation v^3 on S, while there is only one valuation v^3 induced by a given coherent valuation $S_j^{\pm}$. Note also that when we write $z^2 \;{}_{S_j^{\pm}}\!\rightleftarrows_S\; v^3$ it is implicit that z^2 is coherent on $S_j^{\pm}$. The content of the following proposition, which is of use in establishing the correctness of the encodings we present in this chapter, should be fairly obvious:

Proposition 3.1.1. Let z_1^2, z_2^2, v_1^3, v_2^3, ..., v_n^3 $(n \geq 1)$ be arbitrary valuations, $S_{j_1}^{\pm}, S_{j_2}^{\pm}, ..., S_{j_n}^{\pm}$ disjoint sets of variables, and Y a set of variables. Then

1. If Y is disjoint with each of $S_{j_i}^{\pm}$ for $1 \leq i \leq n$, z_1^2 is coherent on each of $S_{j_i}^{\pm}$, and $z_1^2 \;{}_{S_{j_i}^{\pm}}\!\rightleftarrows_S\; v_i^3$ for each i $(1 \leq i \leq n)$, then $z_3^2 := z_1^2[Y/z_2^2(Y)]$ is also coherent on each of $S_{j_i}^{\pm}$, and it also holds that $z_3^2 \;{}_{S_{j_i}^{\pm}}\!\rightleftarrows_S\; v_i^3$ for each i.

2. If $S_{j_i}^{\pm} \subseteq Y$, z_2^2 is coherent on $S_{j_i}^{\pm}$, and $z_2^2 \;{}_{S_{j_i}^{\pm}}\!\rightleftarrows_S\; v_i^3$ for each i $(0 \leq i \leq n)$, then $z_3^2 := z_1^2[Y/z_2^2(Y)]$ is also coherent on each $S_{j_i}^{\pm}$ and it also holds that $z_3^2 \;{}_{S_{j_i}^{\pm}}\!\rightleftarrows_S\; v_i^3$ for each i.

Proof. (sketch) Item one follows from the fact that z_1^2 and z_3^2 agree on the variables in each of the $S_{j_i}^{\pm}$s for $1 \leq i \leq n$. Item two from the fact that z_2^2 and z_3^2 agree on the variables in each the $S_{j_i}^{\pm}$s. □

A seemingly complicated yet, on closer inspection, immediate corollary of this proposition is the following:

Corollary 3.1.1. Let z_1^2, z_2^2, $v_{j_1}^3$, $v_{j_2}^3$, ..., $v_{j_n}^3$,$v_{k_1}^3$, $v_{k_2}^3$, ..., $v_{k_m}^3$ $(n \geq 1,\ m \geq 1)$ be arbitrary valuations, $S_{j_1}^{\pm}, S_{j_2}^{\pm}, ..., S_{j_n}^{\pm}$, $S_{k_1}^{\pm}, S_{k_2}^{\pm}, ..., S_{k_m}^{\pm}$ disjoint sets of variables and $Y \supseteq S_{k_1}^{\pm} \uplus S_{k_2}^{\pm} \uplus ... \uplus S_{k_m}^{\pm}$ a set of variables that is disjoint with each $S_{j_i}^{\pm}$ for $1 \leq i \leq n$. Assume also on the one hand z_1^2 is coherent on each of $S_{j_i}^{\pm}$, and $z_1^2 \;{}_{S_{j_i}^{\pm}}\!\rightleftarrows_S\; v_{j_i}^3$ for each i such that $1 \leq i \leq n$.

On the other hand assume also that z_2^2 is coherent on each of $S^{\pm}_{k_i}$, and $z_3^2 \; {}_{S^{\pm}_{k_i}}\!\!\rightleftarrows_S \; v^3_{k_i}$ for each i ($1 \leq i \leq m$). Then, in the first place, $z_3^2 = z_1^2[Y/z_2^2(Y)]$ is coherent on each $S^{\pm}_{j_i}$ for $1 \leq i \leq n$ and also on each $S^{\pm}_{k_i}$ for $1 \leq i \leq m$. In the second place, it also holds that $z_3^2 \; {}_{S^{\pm}_{j_i}}\!\!\rightleftarrows_S \; v^3_{j_i}$ for $1 \leq i \leq n$ and also $z_3^2 \; {}_{S^{\pm}_{k_i}}\!\!\rightleftarrows_S \; v^3_{k_i}$ for $1 \leq i \leq m$.

Proof. (sketch) Follows directly from Proposition 3.1.1 because the assumptions of both items of this proposition are satisfied. □

A valuation z^2 is coherent on some set of variables $S^{\pm}_j$ for some $j \geq 1$ if and only if it satisfies the following formula:

$$coh_j[S] := \bigwedge_{s \in S} \neg(s^{\oplus}_j \wedge s^{\ominus}_j)$$

as is stated formally in the following proposition:

Proposition 3.1.2. A valuation z^2 is coherent on $S^{\pm}_j$ ($j \geq 1$) if and only if $z^2 \models coh_j[S]$.

Proof. A valuation z^2 is coherent on $S^{\pm}_j$ (according to Definition 3.1.1) if and only if for no $s \in S$ it is the case that $z^2(s^{\oplus}_j) = 1$ and $z^2(s^{\ominus}_j) = 1$ if and only if for each $s \in S$, $z^2(\neg(s^{\oplus}_j \wedge s^{\ominus}_j)) = 1$ if and only if (by semantics and definition of $coh_j[S]$) $z^2(\bigwedge_{s \in S} \neg(s^{\oplus}_j \wedge s^{\ominus}_j)) = z^2(coh_j[S]) = 1$. □

In the encodings we present in this section, we also need to be able to express via QBFs that a certain acceptance condition is evaluated to 1, 0 or $\frac{1}{2}$ by a three valued valuation on an ADF in accordance with the semantics of Kleene's strong three valued logic. We do this by defining a function $val_j(\phi, x)$ where ϕ is a propositional formula and $x \in \{0, 1, \frac{1}{2}\}$, such that for valuations v^3_j and z^2 such that $z^2 \; {}_{S^{\pm}_j}\!\!\rightleftarrows_S \; v^3_j$, $v^3_j(\phi) = x$ if and only if $z^2(val_j(\phi, x)) = 1$. The function val_j (for any $j \geq 1$), in turn, depends on the following functions τ^1_j and τ^2_j:

Definition 3.1.3. For a propositional variable s and propositional formulas ψ, ϕ with variables in a set S, the functions τ^1_j and τ^2_j (for any $j \geq 1$) are defined as follows:

- $\tau^1_j(\top) = \top$; $\tau^2_j(\top) = \bot$
- $\tau^1_j(\bot) = \bot$; $\tau^2_j(\bot) = \top$
- $\tau^1_j(s) = s^{\oplus}_j$; $\tau^2_j(s) = s^{\ominus}_j$
- $\tau^1_j(\neg\psi) = \tau^2_j(\psi)$; $\tau^2_j(\neg\psi) = \tau^1_j(\psi)$
- $\tau^1_j(\psi \wedge \phi) = \tau^1_j(\psi) \wedge \tau^1_j(\phi)$; $\tau^2_j(\psi \wedge \phi) = \tau^2_j(\psi) \vee \tau^2_j(\phi)$
- $\tau^1_j(\psi \vee \phi) = \tau^1_j(\psi) \vee \tau^1_j(\phi)$; $\tau^2_j(\psi \vee \phi) = \tau^2_j(\psi) \wedge \tau^2_j(\phi)$

For these functions τ^1_j and τ^2_j the following holds:

Lemma 3.1.1. For valuations z^2 and v^3_j such that $z^2 \;{}_{S^{\pm}_j}\!\rightleftarrows_{S}\; v^3_j$ and any propositional formula ϕ with variables in S, $v^3_j(\phi) = 1$ if and only if $z^2(\tau^1_j(\phi)) = 1$, and $v^3_j(\neg\phi) = 1$ if and only if $z^2(\tau^2_j(\phi)) = 1$.

Proof. The proof is by induction on the structure of the formula ϕ.

Consider first the base case $\phi = \top$. For every valuation v^3_j, $v^3_j(\top) = 1$ is the case and for every valuation z^2, $z^2(\tau^1_j(\top)) = z^2(\top) = 1$ holds. Also, for no valuation v^3_j, $v^3_j(\neg\top) = 1$ is the case and for no valuation z^2, $z^2(\tau^2_j(\top)) = z^2(\bot) = 1$ can hold. So the proposition holds trivially for $\phi = \top$.

The case $\phi = \bot$ can be proved using the same line of argument as for $\phi = \top$.

Consider now the base case $\phi = s$ for some $s \in S$. One has $v^3_j(s) = 1$ if and only if (by Definition 3.1.2) $z^2(s^{\oplus}_j) = 1$ and $z^2(s^{\ominus}_j) = 0$ if and only if (since z^2 is coherent on $S^{\pm}_j$, $z^2(s^{\ominus}_j) = 1$ can not be the case) $z^2(s^{\oplus}_j) = 1$ if and only if (by the definition of τ^1_j) $z^2(\tau^1_j(s)) = 1$. Using the same line of argument, it is easy to prove that $v^3_j(\neg s) = 1$ if and only if $z^2(\tau^2_j(s)) = 1$.

Consider now the inductive step for $\phi = \neg\psi$ for some formula ψ.

One has $v^3_j(\neg\psi) = 1$ if and only if (by inductive hypothesis) $z^2(\tau^2_j(\psi)) = 1$ if and only if (by the definition of τ^1_j) $z^2(\tau^1_j(\neg\psi)) = 1$. Also $v^3_j(\neg\neg\psi) = 1$ if and only if (by semantics) $v^3_j(\psi) = 1$ if and only if (by inductive hypothesis) $z^2(\tau^1_j(\psi)) = 1$ if and only if (by the definition of τ^2_j) $z^2(\tau^2_j(\neg\psi)) = 1$. In conclusion, the proposition holds for the case $\phi = \neg\psi$.

The inductive step for the cases $\phi = \psi \wedge \rho$ and $\phi = \psi \vee \rho$ for formulas ψ and ρ are proved in a similar manner as for the case $\phi = \neg\psi$. $\square$

We now give the definition of the before mentioned function val_j (for each $j \geq 1$):

Definition 3.1.4. For a propositional formula ϕ with variables in S, the function $val_j(\phi, x)$ ($j \geq 1$) for $x \in \{0, 1, \frac{1}{2}, \infty\}$ is defined as follows:

- $val_j(\phi, 1) = \tau^1_j(\phi) \wedge \neg\tau^2_j(\phi)$
- $val_j(\phi, 0) = \neg\tau^1_j(\phi) \wedge \tau^2_j(\phi)$
- $val_j(\phi, \frac{1}{2}) = \neg\tau^1_j(\phi) \wedge \neg\tau^2_j(\phi)$
- $val_j(\phi, \infty) = \tau^1_j(\phi) \wedge \tau^2_j(\phi)$

Here $val_j(\phi, \infty)$ expresses the fact that under some valuation v^3_j, ϕ is both "true" and "false", which in fact implies that any two valued valuation that satisfies this formula is inconsistent on $S^{\pm}_j$. Note, in particular, that now $coh_j[S]$ can be expressed as

$$\textstyle\bigwedge_{s \in S} \neg val_j(s, \infty).$$

The above mentioned desired result is stated formally as follows:

Proposition 3.1.3. For valuations z^2 and v^3_j such that $z^2 \;{}_{S^{\pm}_j}\!\rightleftarrows_S\; v^3$ and every propositional formula ϕ (with all variables occurring in S), $z^2(val_j(\phi, x)) = 1$ if and only if $v^3_j(\phi) = x$ for $x \in \{1, 0, \frac{1}{2}\}$.

Proof. We give the proof of the case for $x = \frac{1}{2}$ here, the proof of the other cases are similar. For this case one has, $z^2(val_j(\phi, \frac{1}{2})) = 1$ if and only if (by Definition 3.1.4) $z^2(\neg\tau^1_j(\phi) \wedge \neg\tau^2_j(\phi)) = 1$ if and only if $z^2(\tau^1_j(\phi)) = 0$ and $z^2(\tau^2_j(\phi)) = 0$ if and only if (by Lemma 3.1.1) v^3_j does not satisfy ϕ nor $\neg\phi$ if and only if (by semantics) $v^3_j(\phi) = \frac{1}{2}$. □

We also often make use the proposition we state next when proving the "correctness" of the encodings we give in this work. It provides some of the reason for which, as we have already hinted at in Section 2.2, using QBFs for expressing statements about propositional valuations is a natural choice:

Proposition 3.1.4. Let ϕ be an arbitrary QBF, ψ a closed QBF, and $P = \{p_1, p_2, ..., p_n\}$ propositional variables. Then

1. If $\phi = \exists P\psi$, then $z^2 \models \phi$ if and only if $z^2[P/Y] \models \psi$ for some $Y = \{x_1, .., x_n\}$ where each $x_i \in \{0, 1\}$.
2. If $\phi = \forall P\psi$, then $z^2 \models \phi$ if and only if $z^2[P/Y] \models \psi$ for any $Y = \{x_1, ..., x_n\}$ where each $x_i \in \{0, 1\}$.

Proof. (sketch) For item one, from the semantics of QBFS (see Definition 2.2.5) it is easy to prove that $z^2 \models \exists P\psi$ if and only if $z^2[P_i/Y_i] \models \exists Q_i\psi$ for some $Y_i = \{x_1, .., x_i\}$ ($x_i \in \{0, 1\}$) by induction on i ($1 \leq i \leq n$) where $P_i := \{p_1, ..., p_i\}$ and $Q_i := P \setminus P_i$. From this item 1 follows directly. Item 2 can be proved in a similar fashion. □

Finally, a simple corollary of this proposition that we also use in the proofs in the following sections is the following reformulation of Proposition 2.2.2:

Corollary 3.1.2. Let ϕ be an arbitrary QBF, ψ a closed QBF, and P a set of propositional variables. Then

1. If $\phi = \exists P\psi$, then ϕ is true if and only if for some z^2, $z^2 \models \psi$.
2. If $\phi = \forall P\psi$, then ϕ is true if and only if for all z^2, $z^2 \models \psi$.

Proof. (sketch)

1. If ϕ is true, then for all z^2, $z^2 \models \phi$ from which by Proposition 3.1.2 it holds that for all z^2 and some Y $z^2[P/Y] \models \psi$. Hence, there also exists some $z'^2 = z^2[P/Y]$ such that $z'^2 \models \psi$. On the other hand, if for some z^2, $z^2 \models \psi$ also for any z'^2, $z'^2[P/z^2(Y)] \models \psi$ holds since z'^2 and z^2 agree on the free variables of ψ. Hence, by Proposition 3.1.2 for all z'^2, $z'^2 \models \phi$.
2. Similar to proof of item 1.

□

3.2 Encodings for three and two valued models

In the following we give the encodings of the reasoning tasks we consider in this work associated to three valued and two valued models of ADFs. As explained at the beginning of this chapter, we do so by first presenting the defining formulas for three and two valued models.

Given an arbitrary ADF $D = (S, C = \{\phi_s\}_{s \in S})$, the following formulas encode the conditions under which a three valued model v_j^3 can assign 1 and 0 respectively to a statement $s \in S$ based on the acceptance condition associated to s in D:

- $\mathit{mod}_j^1[D](s) := \mathit{val}_j(s, 1) \rightarrow \mathit{val}_j(\phi_s, 1)$
- $\mathit{mod}_j^0[D](s) := \mathit{val}_j(s, 0) \rightarrow \mathit{val}_j(\phi_s, 0)$

The following formula then encodes the fact that a three valued model must satisfy the above conditions for all statements $s \in S$:

$$\mathit{m\text{-}cond}_j[D] := \textstyle\bigwedge_{s \in S}(\mathit{mod}_j^1[D](s) \wedge \mathit{mod}_j^0[D](s))$$

Stated more formally:

Lemma 3.2.1. Let $D = (S, C = \{\phi_s\}_{s \in S})$ be an ADF. Then

1. If $z^2 \models \mathit{m\text{-}cond}_j[D]$ and z^2 is coherent on $S_j^{\pm}$, then v_j^3 such that $z^2 \ {}_{S_j^{\pm}}{\rightleftarrows}_{S}\ v_j^3$ is a three valued model of D.
2. If v_j^3 is a three valued model of D, then for any z^2 such that $z^2 \ {}_{S_j^{\pm}}{\rightleftarrows}_{S}\ v_j^3$ it holds that $z^2 \models \mathit{m\text{-}cond}_j[D]$.

Proof.

1. Assume that $z^2 \models \mathit{m\text{-}cond}_j[D]$ and z^2 is coherent on $S_j^{\pm}$. Consider now v_j^3 such that $z^2 \ {}_{S_j^{\pm}}{\rightleftarrows}_{S}\ v_j^3$ and an arbitrary $s \in S$. If $v_j^3(s) = 1$, then by Proposition 3.1.3 $z^2(\mathit{val}_j(s, 1)) = 1$ and, hence, since $z^2 \models \mathit{m\text{-}cond}_j[D] = \bigwedge_{s \in S}(\mathit{mod}_j^1[D](s) \wedge \mathit{mod}_j^0[D](s))$, i.e. in particular $z^2 \models \mathit{mod}_j^1[D](s) = \mathit{val}_j(s, 1) \rightarrow \mathit{val}_j(\phi_s, 1)$, then also $z^2(\mathit{val}_j(\phi_s, 1)) = 1$ must be the case. Hence, by Proposition 3.1.3, $v_j^3(\phi_s) = 1$. By the same line of argument, if $v_j^3(s) = 0$, since $z^2 \models \mathit{mod}_j^0[D](s) = \mathit{val}_j(s, 0) \rightarrow \mathit{val}_j(\phi_s, 0)$, $v_j^3(\phi_s) = 0$ must be the case. In conclusion, since s was arbitrary, if $v_j^3(s) \neq \frac{1}{2}$, one has that $v_j^3(s) = v_j^3(\phi_s)$ for any $s \in S$, i.e. v_j^3 is a three valued model of D.
2. Assume that v_j^3 is a three valued model of D. Consider an arbitrary $s \in S$ and assume that for any z^2 such that $z^2 \ {}_{S_j^{\pm}}{\rightleftarrows}_{S}\ v_j^3$, $z^2(\mathit{val}_j(s, 1)) = 1$. By Proposition 3.1.3 this means that $v_j^3(s) = 1$ and, hence, since v_j^3 is a three valued model of D, $v_j^3(\phi_s) = 1$ which, by Proposition 3.1.3 implies that $z^2(\mathit{val}_j(\phi_s, 1)) = 1$. In conclusion, $z^2 \models \mathit{mod}_j^1[D](s) = \mathit{val}_j(s, 1) \rightarrow \mathit{val}_j(\phi_s, 1)$. By the same line of argument, starting with the assumption that

$z^2(val_j(s,0)) = 1$ one reaches the conclusion that $z^2 \models mod_j^0[D](s) = val_j(s,0) \rightarrow val_j(\phi_s, 0)$. Since $s \in S$ was arbitrary, $z^2 \models mod_j^1[D](s)$ and $z^2 \models mod_j^0[D](s)$ for every $s \in S$ and, hence, $z^2 \models \bigwedge_{s \in S}(mod_j^1[D](s) \wedge mod_j^0[D](s)) = m\text{-}cond_j[D]$.

□

As can be expected from the assumptions of the previous lemma, for an arbitrary ADF $D = (S, C = \{\phi_s\}_{s \in S})$ the following formula now encodes that v_j^3 is a three valued model for D:

$$3mod_j[D] := coh_j[S] \wedge m\text{-}cond_j[D]$$

The following lemma states this fact:

Lemma 3.2.2. Let $D = (S, C = \{\phi_s\}_{s \in S})$ be an ADF. Then

1. If $z^2 \models 3mod_j[D]$ then z^2 is coherent on $S_j^{\pm}$ and v_j^3 such that $z^2 \;{}_{S_j^{\pm}}{\rightleftarrows}_{S}\; v_j^3$ is a three valued model of D.
2. If v_j^3 is a three valued model of D, then for any z^2 such that $z^2 \;{}_{S_j^{\pm}}{\rightleftarrows}_{S}\; v_j^3$ it holds that $z^2 \models 3mod_j[D]$.

Proof.

1. Assume that $z^2 \models 3mod_j[D] = coh_j[S] \wedge m\text{-}cond_j[D]$. Then $z^2 \models coh_j[S]$ from which by Proposition 3.1.2 it follows that z^2 is coherent on $S_j^{\pm}$. Now, since also $z^2 \models m\text{-}cond_j[D]$ by Lemma 3.2.1 v_j^3 such that $z^2 \;{}_{S_j^{\pm}}{\rightleftarrows}_{S}\; v_j^3$ is a three valued model of D.
2. If v_j^3 is a three valued model of D, then any z^2 such that $z^2 \;{}_{S_j^{\pm}}{\rightleftarrows}_{S}\; v_j^3$ is by this assumption coherent and, hence, by Proposition 3.1.2, $z^2 \models coh_j[S]$. Now by Lemma 3.2.1 also $z^2 \models m\text{-}cond_j[D]$ and, hence, $z^2 \models coh_j[S] \wedge m\text{-}cond_j[D] = 3mod_j[D]$.

□

Two valued models of an ADF $D = (S, C = \{\phi_s\}_{s \in S})$ are a special case of three valued models, since they are three valued models for which no statement is assigned to $\frac{1}{2}$. This can be encoded via the following formula:

$$\neg\tfrac{1}{2}_j[S] := \bigwedge_{s \in S} \neg val_j(s, \tfrac{1}{2})$$

as is stated formally in the next lemma:

Lemma 3.2.3. Let $D = (S, C = \{\phi_s\}_{s \in S})$ be an ADF. Then

1. If $z^2 \models \neg\frac{1}{2}_j[S]$ and z^2 coherent on $S_j^{\pm}$, then v_j^3 such that $z^2 \;{}_{S_j^{\pm}}{\rightleftarrows}_{S}\; v_j^3$ is two valued.

2. If v^3_j is two valued, then for any z^2 such that $z^2 \;_{S^{\pm}_j}\!\rightleftarrows_S v^3_j$ it holds that $z^2 \models \neg \frac{1}{2}_j[S]$.

Proof.

1. Assume that $z^2 \models \neg\frac{1}{2}_j[S] = \bigwedge_{s \in S} \neg val_j(s, \frac{1}{2})$ and z^2 is coherent under $S^{\pm}_j$. Then $z^2 \models \neg val_j(s, \frac{1}{2})$ for every $s \in S$, i.e. there exists no $s \in S$ for which $z^2 \models val_j(s, \frac{1}{2})$ which by Proposition 3.1.3 means that there exists no $s \in S$ such that $v^3_j(s) = \frac{1}{2}$ for v^3_j such that $z^2 \;_{S^{\pm}_j}\!\rightleftarrows_S v^3_j$. This in turn means that v^3_j is two valued.

2. Assume that v^3_j is two valued, i.e. there exists no $s \in S$ such that $v^3_j(s) = \frac{1}{2}$. Then, by Proposition 3.1.3 for any z^2 such that $z^2 \;_{S^{\pm}_j}\!\rightleftarrows_S v^3_j$ there exists no $s \in S$ such that $z^2 \models val_j(s, \frac{1}{2})$. Hence, $z^2 \models \bigwedge_{s \in S} \neg val_j(s, \frac{1}{2}) = \neg\frac{1}{2}_j[S]$.

$\square$

The following formula now encodes that a valuation v^3_j is a two-valued model for D:

$$2mod_j[D] := coh_j[S] \wedge \neg\frac{1}{2}_j[S] \wedge m\text{-}cond_j[D]$$

The next lemma, whose proof is similar to the proof of Lemma 3.2.2 (hence we do not include the proof here) states this formally:

Lemma 3.2.4. Let $D = (S, C = \{\phi_s\}_{s \in S})$ be an ADF. Then

1. If $z^2 \models 2mod_j[D]$ then z^2 is coherent on $S^{\pm}_j$ and v^3_j such that $z^2 \;_{S^{\pm}_j}\!\rightleftarrows_S v^3_j$ is a two valued model of D.

2. If v^3_j is a two valued model of D, then for any z^2 such that $z^2 \;_{S^{\pm}_j}\!\rightleftarrows_S v^3_j$ it holds that $z^2 \models 2mod_j[D]$.

From Lemmas 3.2.2 and 3.2.4 and the correspondence between associated valuations for QBFs and ADFs, it follows that given an ADF D, the formulas

$$3mod_1[D]$$

and

$$2mod_1[D]$$

provide faithful encodings of ENUM$_{3mod}$ and ENUM$_{2mod}$ respectively into QENUM. These are clearly also polynomial, in fact linear in the size of D.

We finally also give the encodings of the various decision problems with respect to three and two valued models in the following propositio. We provide encodings only for those tasks which are non-trivial in regards to their complexity (see Section 2.4). We remind the reader also that EXISTS$_{2mod}$ and EXISTS$^{\neg\emptyset}_{2mod}$ are equivalent.

Proposition 3.2.1. Given arbitrary ADF $D = (S, C = \{\phi_s\}_{s \in S})$, the following hold:

1. (EXISTS$^{\neg\emptyset}_{3mod}$) D has a three valued model v^3 such that $v^3(s) = 1$ for some $s \in S$ if and only if $\exists S_1^{\pm}(3mod_1[D] \wedge (\bigvee_{s \in S} \neg val_1(s, \frac{1}{2})))$ is true.

2. (CRED$_{3mod}$) For an arbitrary $s_* \in S$, D has a three valued model v^3 such that $v^3(s_*) = 1$ if and only if $\exists S_1^{\pm}(3mod_1[D] \wedge val_1(s_*, 1))$ is true.

3. (EXISTS$_{2mod}$ / EXISTS$^{\neg\emptyset}_{2mod}$) D has a (non-trivial) two valued model if and only if $\exists S_1^{\pm} 2mod_1[D]$ is true.

4. (CRED$_{2mod}$) For an arbitrary $s_* \in S$, D has a two valued model v^3 such that $v^3(s_*) = 1$ if and only if $\exists S_1^{\pm}(2mod_1[D] \wedge val_1(s_*, 1))$ is true.

5. (SKEPT$_{2mod}$) For an arbitrary $s_* \in S$, for all two valued models v^3 of D $v^3(s_*) = 1$ holds if and only if $\forall S_1^{\pm}(2mod_1[D] \rightarrow val_1(s_*, 1))$ is true.

Proof. We only give the proof of item 2 here as an example, the proof of the other items are similar.

Assume first that CRED$_{3mod}(s_*) = \exists S_1^{\pm}(3mod_1[D] \wedge val_1(s_*, 1))$ is true. By Corollary 3.1.2 this means that there exists a z^2 such that $z^2 \models 3mod_1[D] \wedge val_1(s_*, 1)$. Hence, $z^2 \models 3mod_1[D]$ which, by Lemma 3.2.2 means that v^3 such that $z^2 \;{}_{S_1^{\pm}}{\rightleftarrows}_{S}\; v^3$ is a three valued model of D. Also, since $z^2 \models (val_1(s_*, 1))$, i.e. $z^2(val_1(s_*, 1)) = 1$ by Proposition 3.1.3 $v^3(s_*) = 1$. In conclusion, D has a three valued model v^3 such that $v^3(s_*) = 1$.

Assume now that D has a three valued model v^3 such that $v^3(s_*) = 1$. Then by Lemma 3.2.2 $z^2 \models 3mod_1[D]$ for some z^2 such that $z^2 \;{}_{S_1^{\pm}}{\rightleftarrows}_{S}\; v^3$. Now, since $v^3(s_*) = 1$, by Proposition 3.1.3, $z^2(val_1(s_*, 1)) = 1$. Hence, $z^2 \models 3mod_1[D] \wedge val_1(s_*, 1)$ and, so, by Corollary 3.1.2, $\exists S_1^{\pm}(3mod_1[D] \wedge val_1(s_*, 1)) =$ CRED$_{3mod}(s_*)$ is true. □

Proposition 3.2.1 gives faithful encodings of all the decision problems with respect to three and two valued models of ADFs we consider in this work. It should also be clear that all the encodings are polynomial in the size of the ADF, in fact they are linear. For all non trivial decision problems except SKEPT$_{2mod}$, the encodings give QBFs of prefix type Σ_1 indicating that the complexity of these reasoning tasks is in NP, while the encoding for SKEPT$_{2mod}$ is a QBF of prefix type Π_1, from which it can be concluded that the reasoning task is in co-NP. Hence, all the encodings indicate complexity. From the results presented in Table 2.2 it can also be concluded that all the encodings for the decision problems are also adequate.

The encodings given for EXISTS$^{\neg\emptyset}_{3mod}$ and CRED$_{3mod}$ in fact deliver part of the proof that these reasoning problems are, as is the case of EXISTS$^{\neg\emptyset}_{2mod}$ and CRED$_{2mod}$, NP complete, reason for which the relevant areas in Table 2.2 refer to this section of our work. Specifically, the encodings indicate, via the correspondece between prefix types of QBFs and the classes of the polynomial hierarchy presented in Section 2.4, that these reasoning problems are in NP. That EXISTS$_{2mod}$ (and, hence, EXISTS$^{\neg\emptyset}_{2mod}$) as well as CRED$_{2mod}$ can be reduced to EXISTS$^{\neg\emptyset}_{3mod}$ and CRED$_{3mod}$ gives NP-hardness of the latter problems via the results regarding two valued models presented in [Strass and Wallner, 2013]. One possible reduction hinges on the following lemma, whose proof, given that complexity analysis is not the main task we have set out to carry out in this work, we only sketch (in some detail) here:

Lemma 3.2.5. Let $D = (S, C = \{\phi_s\}_{s \in S})$ be an ADF. Then v^3 is a (non-trivial) two valued model of D if and only if v^3 is a non-trivial three valued model of the ADF $D' = (S, \{\phi'_s\}_{s \in S})$ where, given an arbitrary ordering $\{s_1, s_2, ..., s_n\}$ of all the statements in S (i.e. $|S| = n$), ϕ'_{s_i} for each s_i for $1 \leq i < n$ is defined as follows:

$$\phi'_{s_i} := (\neg s_i \vee (\phi_{s_i} \wedge (s_{i+1} \vee \neg s_{i+1}))) \wedge (s_i \vee \phi_{s_i} \vee \neg(s_{i+1} \vee \neg s_{i+1}))$$

and

$$\phi'_{s_n} := (\neg s_n \vee (\phi_{s_n} \wedge (s_1 \vee \neg s_1))) \wedge (s_n \vee \phi_{s_n} \vee \neg(s_1 \vee \neg s_1))$$

Proof. (sketch) Assume first that v^3 is a (non-trivial) two valued model of D. Then in the first place, it is clear it is also non-trivial for D'. Also, since v^3 is two valued, for each s_i ($1 \leq i \leq n$), $v^3(s_i \vee \neg s_i) = 1$ (FACT A). Also, given an arbitrary s_i in S, if $v^3(s_i) = 1$, then clearly also $v^3(s_i \vee \phi_{s_i} \vee \neg(s_{i+1} \vee \neg s_{i+1})) = 1$. Moreover, since $v^3(\phi_{s_i}) = 1$ (v^3 is a model) and FACT A, also $v^3(\neg s_i \vee (\phi_{s_i} \wedge (s_{i+1} \vee \neg s_{i+1}))) = 1$. In conclusion, $v^3(\phi'_{s_i}) = 1$. If, on the other hand, $v^3(s_i) = 0$, then, since also $v^3(\phi_{s_i}) = 0$ and (because of FACT A) $v^3(\neg(s_{i+1} \vee \neg s_{i+1})) = 0$, $v^3(s_i \vee \phi_{s_i} \vee \neg(s_{i+1} \vee \neg s_{i+1})) = 0$ holds and, hence, $v^3(\phi'_{s_i}) = 0$.

Assume now that v^3 is a non-trivial three valued model of D'. Then, in the first place, clearly v^3 is also non-trivial for D. In the second place, for any $s_i \in S$ ($1 \leq i < n$), if $v^3(s_i) = 1$, then because $v^3(\phi'_{s_i}) = 1$, in particular, $v^3(\neg s_i \vee (\phi_{s_i} \wedge (s_{i+1} \vee \neg s_{i+1}))) = 1$ which, since $v^3(\neg s_i) = 0$ implies that $v^3(\phi_{s_i}) = 1$. On the other hand, if $v^3(s_i) = 0$, then since $v^3(\neg s_i) = 1$, $v^3(\neg s_i \vee (\phi_{s_i} \wedge (s_{i+1} \vee \neg s_{i+1}))) = 1$ holds. Therefore, because $v^3(\phi'_{s_i})$ = 0 (v^3 is a model), $v^3(s_i \vee \phi_{s_i} \vee \neg(s_{i+1} \vee \neg s_{i+1})) = 0$ must hold from, which, in particular it can be concluded that $v^3(\phi_{s_i}) = 0$. By the same line of reasoning, also if $v^3(s_n) = x$ then $v^3(\phi_{s_n}) = x$ for $x \in \{1, 0\}$. Finally, using essentially the same reasoning by induction on the position of a statment in an order $\{s_j, s_{j+1}, .., s_n, s_1, .., s_{j-1}\}$ such that $v^3(s_j) = 1$ or $v^3(s_j) = 0$ ($1 \leq j \leq n$) it can be relatively easily be proved that for every s_i ($1 \leq i \leq n$), $v^3(s_i \vee \neg s_i) = 1$. From this the fact that v^3 is two valued follows immediately via the semantics of Kleene's strong three valued logic. □

Proposition 3.2.2. EXISTS$^{\neg\emptyset}_{3mod}$ and CRED$_{3mod}$ are NP-complete.

Proof. EXISTS$^{\neg\emptyset}_{3mod}$ and CRED$_{3mod}$ are in NP because of Proposition 3.2.1 and Theorem 2.4.1. By Lemma 3.2.5 an arbitrary v^3 is a (non-trivial) two valued model of an ADF $D = (S, C)$ if and only it is a non-trivial three valued model of D' as defined in the lemma. Hence, in particular, D has a (non-trivial) two valued model if and only if D' does. Moreover, since for any $s_* \in S$ a valuation v^3 such that $v^3(s_*) = 1$ is non-trivial, also D has a two valued model such that $v^3(s_*) = 1$ if and only if D' does. In conclusion $\mathcal{R}_1$ such that

$$\mathcal{R}_1(D) = D'$$

and $\mathcal{R}_2$ such that

$$\mathcal{R}_2(D, s_*) = (D', s_*)$$

for an arbitrary ADF $D = (S, C)$ and $s_* \in S$ are reductions from the decision problem EXISTS$_{2mod}$ to EXISTS$^{\neg\emptyset}_{3mod}$ and CRED$_{2mod}$ to CRED$_{3mod}$ respectively. Since D' can be constructed in polynomial (in fact linear) time from D, it follows, from the fact that the problems EXISTS$_{2mod}$ and CRED$_{2mod}$ are NP hard [Strass and Wallner, 2013], EXISTS$^{\neg\emptyset}_{3mod}$ and CRED$_{3mod}$ are NP-complete. $\square$

3.3 Encodings for admissible valuations

In the following we present the encodings of the reasoning tasks we consider in this work associated to admissible valuations for ADFs. For a valuation v^3, let

$$\mathcal{X}_{v^3} := \{u^3 \mid u^3(s) \leq_i w^3(\phi_s) \text{ for all } s \in S \text{ and for all } w^3 \in [v^3]_2\}$$

The defining formula for admissible valuations is based on the fact that a valuation v^3 is admissible for an ADF $D = (S, \{\phi_s\}_{s\in S})$ if and only if $v^3(s) \leq_i w^3(\phi_s)$ for every $s \in S$ and $w^3 \in [v^3]_2$, i.e. $v^3 \in \mathcal{X}_{v^3}$. For the proof of this we make use of the following proposition which we will also make use of for the encodings for complete valuations in Section 3.4:

Proposition 3.3.1. Let $D = (S, C = \{\phi_s\}_{s\in S})$ be an ADF and v^3 a valuation for D. Then

$$\Gamma_D(v^3) = \sqcup\mathcal{X}_{v^3} = \sqcup\{u^3 \mid u^3(s) \leq_i w^3(\phi_s) \text{ for all } s \in S \text{ and for all } w^3 \in [v^3]_2\}$$

Proof. Note first that $\mathcal{X}_{v^3}$ is not empty since the valuation mapping everything to 0 is in $\mathcal{X}_{v^3}$. Remember now that for an arbitrary $s \in S$,

(i) $\sqcap\{w^3(\phi_s) \mid w^3 \in [v^3]_2\} \leq_i w'^3(\phi_s)$ for all $w'^3 \in [v^3]_2$ and

(ii) if $x \leq_i w^3(\phi_s)$ for all $w^3 \in [v^3]_2$, then $x \leq_i \sqcap\{w^3(\phi_s) \mid w^3 \in [v^3]_2\}$ ($x \in \{1, 0, \frac{1}{2}\}$).

Consider now an arbitrary $u^3 \in \mathcal{X}_{v^3}$. Then $u^3(s) \leq_i w^3(\phi_s)$ for all $s \in S$ and for all $w^3 \in [v^3]_2$. Hence by (ii), $u^3(s) \leq_i \sqcap\{w^3(\phi_s) \mid w^3 \in [v^3]_2\} = \Gamma_D(v^3)(s)$ for all $s \in S$, i.e. $u^3 \leq_i \Gamma_D(v^3)$. Since u^3 is arbitrary, $\Gamma_D(v^3)$ is an upper bound for $\mathcal{X}_{v^3}$.

Now, it is also the case that $\Gamma_D(v^3) \in \mathcal{X}_{v^3}$ since for all $s \in S$, by (i), $\Gamma_D(v^3)(s) = \sqcap\{w^3 \mid w^3 \in [v^3]_2\} \leq_i w'^3(\phi_s)$ for all $w'^3 \in [v^3]_2$.

In conclusion, since $\Gamma_D(v^3)$ is an upper bound for $\mathcal{X}_{v^3}$ and also $\Gamma_D(v^3) \in \mathcal{X}_{v^3}$, $\Gamma_D(v^3) = \sqcup\mathcal{X}_{v^3}$ (In effect, assume that there exists a v'^3 that is an upper bound for $\mathcal{X}_{v^3}$; then, since $\Gamma_D(v^3) \in \mathcal{X}_{v^3}$, it also holds that $\Gamma_D(v^3) \leq_i v'^3$, i.e. $\Gamma_D(v^3)$ is the least upper bound.). $\square$

Now the above mentioned fact is stated formally in the next proposition:

Proposition 3.3.2. Let $D = (S, C = \{\phi_s\}_{s\in S})$ be an ADF and v^3 a three valued valuation on S. Then v^3 is an admissible valuation for D if and only if $v^3(s) \leq_i w^3(\phi_s)$ for every $s \in S$ and $w^3 \in [v^3]_2$.

Proof. Assume first that v^3 is an admissible valuation for D. By definition of admissible valuations this means that $v^3 \leq_i \Gamma_D(v^3)$, i.e. by the definition of $\leq_i$ on valuations, definition of Γ_D, and the definition of $\sqcap$, $v^3(s) \leq_i \Gamma_D(v^3)(s) = \sqcap\{w'^3(\phi_s) \mid w'^3 \in [v^3]_2\} \leq_i w^3(\phi_s)$ for each $s \in S$ and $w^3 \in [v^3]_2$ as desired.

Assume now that $v^3(s) \leq_i w^3(\phi_s)$ for every $s \in S$ and $w^3 \in [v^3]_2$. This means that $v^3(s) \in \mathcal{X}_{v^3}$ and, hence, by Proposition 3.3.1, $v^3 \leq_i \Gamma_D(v^3)$, i.e. v^3 is admissible. □

As for three and two valued models, we construct the defining formula for admissible valuations based on Proposition 3.3.2 in piecemeal fashion. First of all, the following formula encodes that $v_j^3(\phi) \leq_i v_k^3(\psi)$ for propositional formulas ϕ and ψ, as well as valuations v_j^3 and v_k^3:

$$\leq_{i\,(j,k)} [\phi,\psi] := (val_j(\phi,1) \rightarrow val_k(\psi,1)) \wedge (val_j(\phi,0) \rightarrow val_k(\psi,0))$$

as is stated more formally in the next lemma:

Lemma 3.3.1. Let $D = (S, C = \{\phi_s\}_{s\in S})$ be an ADF and ϕ and ψ propositional formulas. Then

1. If $z^2 \models \leq_{i\,(j,k)} [\phi,\psi]$ and z coherent on $S_j^{\pm}$ and $S_k^{\pm}$, then for v_j^3 and v_k^3 such that $z^2 \;{}_{S_j^{\pm}}\!\rightleftarrows_S\; v_j^3$ and $z^2 \;{}_{S_k^{\pm}}\!\rightleftarrows_S\; v_k^3$ it holds that $v_j^3(\phi) \leq_i v_k^3(\psi)$.

2. If $v_j^3(\phi) \leq_i v_k^3(\psi)$, then for any z such that $z^2 \;{}_{S_j^{\pm}}\!\rightleftarrows_S\; v_j^3$ and $z^2 \;{}_{S_k^{\pm}}\!\rightleftarrows_S\; v_k^3$ it holds that $z^2 \models \leq_{i\,(j,k)} [\phi,\psi]$.

Proof.

1. Assume that $z^2 \models \leq_{i\,(j,k)} [\phi,\psi] = (val_j(\phi,1) \rightarrow val_k(\psi,1)) \wedge (val_j(\phi,0) \rightarrow val_k(\psi,0))$ and that z^2 is coherent on $S_j^{\pm}$ and $S_k^{\pm}$. Consider now that $v_j^3(\phi) = 1$ for v_j^3 such that $z^2 \;{}_{S_j^{\pm}}\!\rightleftarrows_S\; v_j^3$. By Proposition 3.1.3 then $z^2 \models val_j(\phi,1)$. Therefore, since $z^2 \models val_j(\phi,1) \rightarrow val_k(\psi,1)$ also $z^2 \models val_k(\psi,1)$ must be the case. Hence, again by Proposition 3.1.3, $v_k^3(\psi) = 1$ must be the case and so $v_j^3(\phi) \leq_i v_k^3(\psi)$. In the same manner, if $v_j^3(\phi) = 0$, since $z^2 \models val_j(\phi,0) \rightarrow val_k(\psi,0)$, $v_k^3(\psi) = 0$ must hold and, hence, also in this case $v_j^3(\phi) \leq_i v_k^3(\psi)$. Finally, if $v_j^3(\phi) = \frac{1}{2}$, then $v_j^3(\phi) \leq_i v_k^3(\psi)$ no matter the value of $v_k^3(\psi)$ by definition of $\leq_i$. In conclusion $v_j^3(\phi) \leq_i v_k^3(\psi)$ holds in general as desired.

2. Assume that $v_j^3(\phi) \leq_i v_k^3(\psi)$. Now if $v_j^3(\phi) = x$ with $x \in \{1,0\}$, since $v_j^3(\phi) \leq_i v_k^3(\psi)$, $v_k^3(\psi) = x$ must hold as well, since 1 and 0 are maximal with respect to $\leq_i$. By Proposition 3.1.3 this means that if $z^2 \models val_j(\phi,x)$, then, $z^2 \models val_k(\psi,x)$ for $x \in \{0,1\}$. In conclusion, $z^2 \models (val_j(\phi,1) \rightarrow val_k(\psi,1)) \wedge (val_j(\phi,0) \rightarrow val_k(\psi,0)) = \leq_{i\,(j,k)} [\phi,\psi]$ as desired.

□

The previous formula can be used in a straightforward manner to encode that for two valuations v_j^3 and v_k^3, $v_j^3(s) \leq_i v_k^3(s)$ for all $s \in S$ for a set of statements S:

$$\leq_{i\,(j,k)}^{\bigwedge} [S] := \bigwedge_{s \in S} \leq_{i\,(j,k)} [s,s]$$

The next lemma states this fact:

Lemma 3.3.2. Let $D = (S, C = \{\phi_s\}_{s \in S})$ be an ADF. Then

1. If $z^2 \models \leq_{i\,(j,k)}^{\bigwedge} [S]$ and z^2 is coherent on $S_j^{\pm}$ and $S_k^{\pm}$, then for v_j^3 and v_k^3 such that $z^2 \;{}_{S_j^{\pm}}\!\rightleftarrows_S\; v_j^3$ and $z^2 \;{}_{S_k^{\pm}}\!\rightleftarrows_S\; v_k^3$ it holds that $v_j^3(s) \leq_i v_k^3(s)$ for all $s \in S$.

2. If $v_j^3(s) \leq_i v_k^3(s)$ for all $s \in S$, then for any z^2 such that $z^2 \;{}_{S_j^{\pm}}\!\rightleftarrows_S\; v_j^3$ and $z^2 \;{}_{S_k^{\pm}}\!\rightleftarrows_S\; v_k^3$ it holds that $z^2 \models \leq_{i\,(j,k)}^{\bigwedge} [S]$.

Proof.

1. Assume that $z^2 \models \leq_{i\,(j,k)}^{\bigwedge} [S] = \bigwedge_{s \in S} \leq_{i\,(j,k)} [s,s]$ and z^2 is coherent on $S_j^{\pm}$ and $S_k^{\pm}$. Then $z^2 \models \leq_{i\,(j,k)} [s,s]$ for each $s \in S$ and so, by Lemma 3.3.1, $v_j^3(s) \leq_i v_k^3(s)$ for each $s \in S$ as desired.

2. Assume that $v_j^3(s) \leq_i v_k^3(s)$ for all $s \in S$. Then, by Lemma 3.3.1, for any z^2 such that $z^2 \;{}_{S_j^{\pm}}\!\rightleftarrows_S\; v_j^3$ and $z^2 \;{}_{S_k^{\pm}}\!\rightleftarrows_S\; v_k^3$ it holds that $z^2 \models \leq_{i\,(j,k)} [s,s]$ for each $s \in S$. Hence, for any such z^2 also $z^2 \models \bigwedge_{s \in S} \leq_{i\,(j,k)} [s,s] = \leq_{i\,(j,k)}^{\bigwedge} [S]$ as desired.

□

The following formula now encodes that a valuation v_j^3 is an extension of another valuation v_k^3 on a set of statements S:

$$ext_{j,k}[S] := coh_j[S] \land \neg \tfrac{1}{2}_j[S] \land \leq_{i\,(k,j)}^{\bigwedge} [S]$$

as is stated formally in the following lemma:

Lemma 3.3.3. Let $D = (S, C = \{\phi_s\}_{s \in S})$ be an ADF. Then

1. If $z^2 \models ext_{j,k}[S]$ and z^2 is coherent on $S_k^{\pm}$, then z^2 is coherent on $S_j^{\pm}$ and for v_j^3 and v_k^3 such that $z^2 \;{}_{S_j^{\pm}}\!\rightleftarrows_S\; v_j^3$ and $z^2 \;{}_{S_k^{\pm}}\!\rightleftarrows_S\; v_k^3$ it holds that $v_j^3 \in [{v_k}^3]_2$.

2. If $v_j^3 \in [{v_k}^3]_2$, then for any z^2 such that $z^2 \;{}_{S_j^{\pm}}\!\rightleftarrows_S\; v_j^3$ and $z^2 \;{}_{S_k^{\pm}}\!\rightleftarrows_S\; v_k^3$ it holds that $z^2 \models ext_{j,k}[S]$.

Proof.

1. Assume that $z^2 \models ext_{j,k}[S] = coh_j[S] \wedge \neg\frac{1}{2}_j[S] \wedge \leq_{i\,(k,j)}^{\bigwedge} [S]$ and z^2 is coherent on $S_k^{\pm}$. Then $z^2 \models coh_j[S]$ and, hence, by Proposition 3.1.2 z^2 is coherent on $S_j^{\pm}$. Since $z^2 \models \neg\frac{1}{2}_j[S]$, by Lemma 3.2.3 v_j^3 such that $z^2 \;{}_{S_j^{\pm}}\!\rightleftarrows_S\; v_j^3$ is a two valued valuation on S. Finally, since also $z^2 \models \leq_{i\,(k,j)}^{\bigwedge} [S]$, by Lemma 3.3.2 also $v_k^3(s) \leq_i v_j^3(s)$ for each $s \in S$ and v_k^3 such that $z^2 \;{}_{S_k^{\pm}}\!\rightleftarrows_S\; v_k^3$. Hence, by definition of an extension, $v_j^3 \in [{v_k}^3]_2$ as desired.
2. Assume $v_j^3 \in [{v_k}^3]_2$. Any z^2 such that $z^2 \;{}_{S_j^{\pm}}\!\rightleftarrows_S\; v_j^3$ and $z^2 \;{}_{S_k^{\pm}}\!\rightleftarrows_S\; v_k^3$ is coherent on $S_j^{\pm}$ and $S_k^{\pm}$. Then, in the first place by Proposition 3.1.2, $z^2 \models coh_j[S]$. Also, v_j^3 is a two valued valuation and, hence, by Proposition 3.2.3 $z^2 \models \neg\frac{1}{2}_j[S]$. Now since also $v_k^3(s) \leq_i v_j^3(s)$ for all $s \in S$, by Lemma 3.3.2 also $z^2 \models \leq_{i\,(k,j)}^{\bigwedge} [S]$. In conclusion, $z^2 \models coh_j[S] \wedge \neg\frac{1}{2}_j[S] \wedge \leq_{i\,(k,j)}^{\bigwedge} [S] = ext_{j,k}[S]$ as desired.

□

The next formula then encodes that for an ADF $D = (S, C = \{\phi_s\}_{s\in S})$ and valuations v_j^3 and v_k^3, $v_j^3(s) \leq_i v_l^3(\phi_s)$ for all $s \in S$ and $v_l^3 \in [{v_k}^3]_2$:

$$\leq_{i\,(j,k,l)}^{\forall ext} [S] := \forall S_l^{\pm}[ext_{l,k}[S] \rightarrow \textstyle\bigwedge_{s\in S} \leq_{i\,(j,l)} [s, \phi_s]]$$

Stated more formally:

Lemma 3.3.4. Let $D = (S, C = \{\phi_s\}_{s\in S})$ be an ADF. Then

1. If $z^2 \models \leq_{i\,(j,k,l)}^{\forall ext} [S]$ and z^2 coherent on $S_j^{\pm}$ and $S_k^{\pm}$, then for v_j^3 and v_k^3 such that $z^2 \;{}_{S_j^{\pm}}\!\rightleftarrows_S\; v_j^3$ and $z^2 \;{}_{S_k^{\pm}}\!\rightleftarrows_S\; v_k^3$ it holds that $v_j^3(s) \leq_i v_l^3(\phi_s)$ for each $s \in S$ and $v_l^3 \in [{v_k}^3]_2$.
2. If $v_j^3(s) \leq_i v_k^3(\phi_s)$ for each $s \in S$ and $v_l^3 \in [{v_k}^3]_2$ then for any z^2 such that $z^2 \;{}_{S_j^{\pm}}\!\rightleftarrows_S\; v_j^3$ and $z^2 \;{}_{S_k^{\pm}}\!\rightleftarrows_S\; v_k^3$ it holds that $z^2 \models \leq_{i\,(j,k,l)}^{\forall ext} [S]$.

Proof.

1. Assume that $z^2 \models \leq_{i\,(j,k,l)}^{\forall ext} [S]$ and z^2 coherent on $S_j^{\pm}$ and $S_k^{\pm}$. Consider now an arbitrary v_l^3 such that $v_l^3 \in [{v_k}^3]_2$ and any z_2^2 such that $z_2^2 \;{}_{S_l^{\pm}}\!\rightleftarrows_S\; v_l^3$. Since $z^2 \models \forall S_l^{\pm}[ext_{l,k}[S] \rightarrow \bigwedge_{s\in S} \leq_{i\,(k,l)} [s, \phi_s]]$ by Proposition 3.1.4 $z^2[S_l^{\pm}/Y] \models ext_{l,k}[S] \rightarrow \bigwedge_{s\in S} \leq_{i\,(j,l)} [s, \phi_s]$ for any Y and, so, in particular, $z_3^2 \models ext_{l,k}[S] \rightarrow \bigwedge_{s\in S} \leq_{i\,(j,l)} [s, \phi_s]$ for $z_3^2 = z^2[S_l^{\pm}/z_2^2(Y)]$. Now, by Corollary 3.1.1 z_3^2 is coherent on $S_j^{\pm}$, $S_k^{\pm}$, and $S_l^{\pm}$. By the same corollary, it is also the case that $z_3^2 \;{}_{S_j^{\pm}}\!\rightleftarrows_S\; v_j^3$, $z_3^2 \;{}_{S_k^{\pm}}\!\rightleftarrows_S\; v_k^3$ and $z_3^2 \;{}_{S_l^{\pm}}\!\rightleftarrows_S\; v_l^3$. Hence, in the first place, since also $v_l^3 \in [{v_k}^3]_2$ by Lemma 3.3.2 it holds that $z_3^2 \models ext_{l,k}[S]$. Now, since $z_3^2 \models ext_{l,k}[S] \rightarrow \bigwedge_{s\in S} \leq_{i\,(j,l)} [s, \phi_s]$, also $z_3^2 \models \bigwedge_{s\in S} \leq_{i\,(j,l)} [s, \phi_s]$ must be the case. Secondly, by Lemma 3.3.2 then also $v_j^3(s) \leq_i v_l^3(\phi_s)$ for all $s \in S$. Since v_l^3 was arbitrary, in conclusion for any $v_l^3 \in [{v_k}^3]_2$, $v_j^3(s) \leq_i v_l^3(\phi_s)$ holds as desired.

2. Assume that $v_j^3 \leq_i v_l^3(\phi_s)$ for each $s \in S$ and v_l^3 in $[{v_k}^3]_2$. Consider now a z^2 such that $z^2 \;{}_{S_j^{\pm}}\!\rightleftarrows_S\; v_j^3$ and $z^2 \;{}_{S_k^{\pm}}\!\rightleftarrows_S\; v_k^3$. Consider also an arbitrary z_2^2 and $z_3^2 = z^2[S_l^{\pm}/z_2^2(S_l^{\pm})]$. By Proposition 3.1.1 now also $z_3^2 \;{}_{S_j^{\pm}}\!\rightleftarrows_S\; v_j^3$ and $z_3^2 \;{}_{S_k^{\pm}}\!\rightleftarrows_S\; v_k^3$ hold. Assume now that $z_3^2 \models ext_{l,k}[S]$. By Lemma 3.3.3 this means that z_3^2 is coherent on $S_l^{\pm}$ and for $v_l'^3$ such that $z_3^2 \;{}_{S_l^{\pm}}\!\rightleftarrows_S\; v_l'^3$ it holds that $v_l'^3 \in [{v_k}^3]_2$. Hence, $v_j^3(s) \leq v_l'^3(\phi_s)$ for each $s \in S$ by assumption and, therefore, by Lemma 3.3.2 also $z_3^2 \models \bigwedge_{s\in S} \leq_{i\,(j,l)} [s, \phi_s]$. In conclusion, $z_3^2 = z[S_l^{\pm}/z_2^2(S_l^{\pm})] \models ext_{l,k}[S] \to \bigwedge_{s\in S} \leq_{i\,(j,l)} [s, \phi_s]$. Since z_2^2 was arbitrary this holds for any such z_2^2 and, therefore, by Proposition 3.1.4 $z^2 \models \forall S_l^{\pm}[ext_{l,k}[S] \to \bigwedge_{s\in S} \leq_{i\,(j,l)} [s, \phi_s]] = \leq_{i\,(j,k,l)}^{\forall ext} [S]$.

□

The following formula finally is the defining formula for admissible valuations of an ADF $D = (S, C = \{\phi_s\}_{s\in S})$:

$$adm_{j,k}[D] := coh_j[S] \wedge \leq_{i\,(j,j,k)}^{\forall ext} [S]$$

This is stated formally in the next lemma:

Lemma 3.3.5. Let $D = (S, C = \{\phi_s\}_{s\in S})$ be an ADF. Then

1. If $z^2 \models adm_{j,k}[D]$ then z^2 is coherent on $S_j^{\pm}$ and v_j^3 such that $z^2 \;{}_{S_j^{\pm}}\!\rightleftarrows_S\; v_j^3$ is an admissible valuation for D.

2. If v_j^3 is an admissible valuation for D, then for any z^2 such that $z^2 \;{}_{S_j^{\pm}}\!\rightleftarrows_S\; v_j^3$ it holds that $z^2 \models adm_{j,k}[D]$.

Proof.

1. Assume that $z^2 \models adm_{j,k}[D] = coh_j[S] \wedge \leq_{i\,(j,j,k)}^{\forall ext} [S]$. Then $z^2 \models coh_j[S]$ and so by Proposition 3.1.2 it follows that z^2 is coherent on $S_j^{\pm}$. Now, since also $z^2 \models \leq_{i\,(j,j,k)}^{\forall ext} [S]$ by Lemma 3.3.4 it follows that $v_j^3(s) \leq_i w^3(\phi_s)$ for each $s \in S$ and $w^3 \in [{v_j}^3]_2$ for v_j^3 such that $z^2 \;{}_{S_j^{\pm}}\!\rightleftarrows_S\; v_j^3$. Finally, by Proposition 3.3.2 it follows that v_j^3 is an admissible valuation for D.

2. Assume that v_j^3 is an admissible valuation for D. Then any z^2 such that $z^2 \;{}_{S_j^{\pm}}\!\rightleftarrows_S\; v_j^3$ is coherent on $S_j^{\pm}$ and, therefore, by Proposition 3.1.2, $z^2 \models coh_j[S]$. Also, since v_j^3 is admissible, by Proposition 3.3.2 it follows that $v_j^3(s) \leq_i w^3(\phi_s)$ for each $s \in S$ and $w^3 \in [{v_j}^3]_2$ and, hence, by Lemma 3.3.4 also $z^2 \models \leq_{i\,(j,j,k)}^{\forall ext} [S]$. In conclusion, since $z^2 \models coh_j[S]$ and $z^2 \models \leq_{i\,(j,j,k)}^{\forall ext} [S]$, also $z^2 \models adm_{j,k}[D] = coh_j[S] \wedge \leq_{i\,(j,j,k)}^{\forall ext} [S]$.

□

From Lemma 3.3.5 (and the correspondence between associated valuations for QBFs and ADFs) it follows that

$$adm_{1,2}[D] := coh_1[S] \wedge \leq^{\forall ext}_{i\,(1,1,2)} [S]$$

provides a faithful encoding of ENUM$_{adm}$ to QENUM. It is clearly also polynomial, in fact linear in the size of the ADF D.

The encodings of the non trivial (see Section 2.4) decision problems with respect to admissible valuations are collected in the following proposition. The proofs are similar as those for Proposition 3.2.1 so we do not include them here.

Proposition 3.3.3. Given arbitrary ADF $D = (S, C = \{\phi_s\}_{s\in S})$, the following hold:

1. (EXISTS$^{\neg\emptyset}_{adm}$) D has an admissible valuation v^3 such that $v^3(s) = 1$ for some $s \in S$ if and only if $\exists S_1^{\pm}(adm_{1,2}[D] \wedge (\bigvee_{s\in S} \neg val_1(s, \frac{1}{2})))$ is true.
2. (CRED$_{adm}$) For an arbitrary $s_* \in S$, D has an admissible valuation v^3 such that $v^3(s_*) = 1$ if and only if $\exists S_1^{\pm}(adm_{1,2}[D] \wedge val_1(s_*, 1))$ is true.

Proof. (sketch) Proof is similar to proof of Proposition 3.2.1 using Proposition 3.1.3, Corollary 3.1.2, and Lemma 3.3.5. □

Proposition 3.3.3 gives faithful encodings of EXISTS$^{\neg\emptyset}_{adm}$ and CRED$_{adm}$. It should also be clear that all the encodings are polynomial in the size of D, in fact they are linear. The following are formulas in prenex normal form equivalent to the corresponding encodings given in Proposition 3.3.3:

1. $\exists S_1^{\pm}\forall S_2^{\pm}(coh_1[S] \wedge [ext_{2,1}[S] \to \bigwedge_{s\in S} \leq_{i\,(1,2)} [s, \phi_s]] \wedge \bigvee_{s\in S} \neg val_1(s, \frac{1}{2}))$
2. $\exists S_1^{\pm}\forall S_2^{\pm}(coh_1[S] \wedge [ext_{2,1}[S] \to \bigwedge_{s\in S} \leq_{i\,(1,2)} [s, \phi_s]] \wedge val_1(s_*, 1))$

These are both formulas of type Σ_2, hence from the results presented in Table 2.2 it can be concluded that the encodings given for EXISTS$^{\neg\emptyset}_{adm}$ and CRED$_{adm}$ are adequate.

3.4 Encodings for complete valuations

In the following we present the encodings of the reasoning tasks we consider in this work associated to complete valuations for ADFs. A valuation is complete for an ADF $D = (S, C = \{\phi_s\}_{s\in S})$ if $v^3 = \Gamma_D(v^3)$. The defining formula for complete valuations is based on Proposition 3.3.1 that states that

$$\Gamma_D(v^3) = \sqcup\mathcal{X}_{v^3} = \sqcup\{u^3 | u^3(s) \leq_i w^3(\phi_s) \text{ for all } w^3 \in [v^3]_2\}.$$

and, hence, v^3 is complete if and only if $v^3 = \sqcup \mathcal{X}_{v^3}$.

It should be clear from the proof of Proposition 3.3.1 that, in fact, $\Gamma_D(v^3)$ is the greatest element of $\mathcal{X}_{v^3}$, i.e. $\Gamma_D(v^3)$ is an upper bound for $\mathcal{X}_{v^3}$ and $\Gamma_D(v^3) \in \mathcal{X}_{v^3}$. That a valuation v_h^3 is an upper bound for $\mathcal{X}_{v_j^3}$ for some valuation v_j^3 can be encoded via the following formula:

$$\text{ub-}\mathcal{X}_{h,j,k,l}[D] := \forall S_k^{\pm}[(coh_k[S] \wedge \leq_{i\,(k,j,l)}^{\forall ext}[S]) \rightarrow \leq_{i\,(k,h)}^{\bigwedge}[S]]$$

as is expressed more precisely in the next lemma:

Lemma 3.4.1. Let $D = (S, C = \{\phi_s\}_{s \in S})$ be an ADF. Then

1. If $z^2 \models$ ub-$\mathcal{X}_{h,j,k,l}[D]$ and z^2 coherent on $S_h^{\pm}$ and $S_j^{\pm}$, then for v_h^3 and v_j^3 such that $z^2 \;{}_{S_h^{\pm}}\!\rightleftarrows_S\; v_h^3$ and $z^2 \;{}_{S_j^{\pm}}\!\rightleftarrows_S\; v_j^3$ it holds that v_h^3 is an upper bound of $\mathcal{X}_{v_j^3}$.

2. If v_h^3 is an upper bound of $\mathcal{X}_{v_j^3}$ then for any z^2 such that $z^2 \;{}_{S_h^{\pm}}\!\rightleftarrows_S\; v_h^3$ and $z^2 \;{}_{S_j^{\pm}}\!\rightleftarrows_S\; v_j^3$ it holds that $z^2 \models$ ub-$\mathcal{X}_{h,j,k,l}[D]$.

Proof. (sketch) Proof is similar to that of Lemma 3.3.4 using Propositions 3.1.2, 3.1.1, 3.1.4, and Lemmas 3.3.2 and 3.3.4. □

Note also that by Proposition 3.3.2, $v_h^3 \in \mathcal{X}_{v_h^3}$ is equivalent to v_h^3 being admissible. In conclusion, v_h^3 is complete if and only if v_h^3 is admissible and an upper bound for $\mathcal{X}_{v_h^3}$. This is encoded in the following formula which is the defining formula we present for complete valuations:

$$comp_{h,j,k,l}[D] := adm_{h,j}[D] \wedge \text{ub-}\mathcal{X}_{h,h,k,l}[D]$$

The following lemma states this fact:

Lemma 3.4.2. Let $D = (S, C = \{\phi_s\}_{s \in S})$ be an ADF. Then

1. If $z^2 \models comp_{h,j,k,l}[D]$ then z^2 is coherent on $S_h^{\pm}$ and v_h^3 such that $z^2 \;{}_{S_h^{\pm}}\!\rightleftarrows_S\; v_h^3$ is a complete valuation for D.

2. If v_h^3 is a complete valuation for D, then for any z^2 such that $z^2 \;{}_{S_h^{\pm}}\!\rightleftarrows_S\; v_h^3$ it holds that $z^2 \models comp_{h,j,k,l}[D]$.

Proof. (sketch) Proof is similar to that of Lemma 3.3.5 using Lemmas 3.3.5 and 3.4.1 and Proposition 3.3.1. □

From Lemma 3.4.2 (and the correspondence between associated valuations for QBFs and ADFs), it follows that

$$comp_{1,2,3,4}[D] := adm_{1,2}[D] \wedge \text{ub-}\mathcal{X}_{1,1,3,4}[D]$$

provides a faithful encoding of ENUM$_{comp}$ to QENUM. It is clearly also polynomial, in fact linear in the size of the ADF D.

Regarding the encodings of the decision problems we consider in this work with respect to complete encodings, we remind the reader first that, as has been explained in Section 2.4, EXISTS$_{comp}$ is trivial. Also, since EXISTS$^{\neg\emptyset}_{comp}$ and CRED$_{comp}$ are equivalent to the problems EXISTS$^{\neg\emptyset}_{adm}$ and CRED$_{adm}$ respectively, the encodings given for the latter two decision problems in Propostion 3.3.3 give adequate encodings of EXISTS$^{\neg\emptyset}_{pref}$ and CRED$_{pref}$. Finally, in Section 3.6 we provide an adequate encoding of SKEPT$_{ground}$ and, hence, since, as has also been explained in Section 2.4, SKEPT$_{comp}$ is equivalent to SKEPT$_{ground}$, this gives us an adequate encoding of SKEPT$_{comp}$ as well.

3.5 Encodings for preferred valuations

In the following we present the encodings of the reasoning tasks we consider in this work associated to preferred valuations for ADFs. The defining formula for preferred valuations is based on a slight reformulation of the definition of preferred valuations. We remind that a valuation v^3 is preferred for an ADF D if v^3 is maximal admissible with respect to $\leq_i$, i.e.

1. v^3 is admissible for D, and

2. there exists no v'^3 that is admissible for D and $v^3 <_i v'^3$ (condition B).

Note that condition B can be reformulated as follows:

"for all v'^3 that are admissible for D, if $v^3 \leq_i v'^3$, then also $v'^3 \leq_i v^3$" (condition B').

In effect, there exists no v'^3 that is admissible (for D) and $v^3 <_i v'^3$ means that for all v'^3 that are admissible, $v^3 \not<_i v'^3$. This in turn is equivalent to for all v'^3 that are admissible, $v^3 \geq_i v'^3$ or $v^3 <>_i v'^3$. This means that for all v'^3 that are admissible, if it is not the case that $v^3 <>_i v'^3$, then $v^3 \geq_i v'^3$ which, in turn, can be reduced to: for all v'^3 that are admissible, if it is not the case that $v^3 <>_i v'^3$ nor that $v^3 >_i v'^3$, then $v^3 = v'^3$. Finally, this can be reformulated as: for all v'^3 that are admissible, if $v^3 \leq_i v'^3$, then also $v'^3 \leq_i v^3$.

Condition B' for a valuation v^3_j being preferred can now be encoded via the following formula:

$$max\text{-}adm_{j,k,l}[D] := \forall S^{\pm}_k [adm_{k,l}[D] \rightarrow (\leq^{\bigwedge}_{i\,(j,k)} [S] \rightarrow \leq^{\bigwedge}_{i\,(k,j)} [S])]$$

as is stated formally in the lemma immediately below.

Lemma 3.5.1. Let $D = (S, C = \{\phi_s\}_{s\in S})$ be an ADF. Then

1. If $z^2 \models max\text{-}adm_{j,k,l}[D]$ and z^2 is coherent on $S^{\pm}_j$, then for v^3_j such that $z^2 \;{}_{S^{\pm}_j}\!\rightleftarrows_S\; v^3_j$ it holds that v^3_j satisfies condition B' (with respect to D).

2. If v_j^3 satisfies condition B' (with respect to D), then for any z^2 such that $z^2 \;{}_{S_j^\pm}\!\rightleftarrows_S\; v_j^3$ it holds that $z^2 \models max\text{-}adm_{j,k,l}[D]$.

Proof. (sketch) Proof is similar to proof of Lemma 3.3.4 using Propositions 3.1.1, 3.1.4, and Lemmas 3.3.2 and 3.3.5. □

The following is, then, the defining formula for preferred valuations:

$$pref_{h,j,k,l}[D] := adm_{h,j}[D] \wedge max\text{-}adm_{h,k,l}[D]$$

as is stated in a formal manner in the next lemma:

Lemma 3.5.2. Let $D = (S, C = \{\phi_s\}_{s \in S})$ be an ADF. Then

1. If $z^2 \models pref_{h,j,k,l}[D]$, then z^2 is coherent on $S_h^\pm$ and v_h^3 such that $z^2 \;{}_{S_h^\pm}\!\rightleftarrows_S\; v_h^3$ is a preferred valuation for D.
2. If v_h^3 is a preferred valuation for D, then for any z^2 such that $z^2 \;{}_{S_h^\pm}\!\rightleftarrows_S\; v_h^3$ it holds that $z^2 \models pref_{h,j,k,l}[D]$.

Proof. (sketch) Proof is similar to proof of Lemma 3.3.5 using Lemmas 3.3.5, 3.5.1, as well as the note about the reformulation of the definition of preferred valuations given at the beginning of this section. □

From Lemma 3.5.2 (and the correspondence between associated valuations for QBFs and ADFs), it then follows that

$$pref_{1,2,3,4}[D] := adm_{1,2}[D] \wedge max\text{-}adm_{1,3,4}[D]$$

provides a faithful encoding of ENUM$_{pref}$ to QENUM. It is clearly also polynomial, in fact linear in the size of the ADF D.

The encoding of SKEPT$_{pref}$, which is the only decision problem with respect to preferred semantics we consider in this work that is not trivial nor can be reduced to corresponding decision problems of other valuation types (see Section 2.4) is given in the following proposition:

Proposition 3.5.1. Given arbitrary ADF $D = (S, C = \{\phi_s\}_{s \in S})$, the following hold:

(SKEPT$_{pref}$) For an arbitrary $s_* \in S$, it holds that $v^3(s_*) = 1$ for all preferred valuations v^3 for D if and only if $\forall S_1^\pm(pref_{1,2,3,4}[D] \rightarrow val_1(s_*, 1))$ is true.

Proof. (sketch) Proof is similar to proof of Proposition 3.2.1 using Proposition 3.1.3, Corollary 3.1.2, and Lemma 3.5.2. □

Proposition 3.5.1 provides a faithful encoding SKEPT$_{pref}$ into QSAT. It should also be clear that the encoding is polynomial in the size of the ADF D, in fact it is linear. The following is the formula in prenex normal form equivalent to the encoding given in Proposition 3.5.1:

$$
\begin{array}{l}
\forall S_1^{\pm} \exists S_2^{\pm} \cup S_3^{\pm} \forall S_4^{\pm} (\\
\qquad coh_1[S] \wedge (ext_{2,1}[S] \rightarrow \bigwedge_{s \in S} \leq_{i\,(1,2)} [s, \phi_s]) \\
\quad \wedge \\
\quad [\\
\qquad [coh_3[S] \wedge (ext_{4,3}[S] \rightarrow \bigwedge_{s \in S} \leq_{i\,(3,4)} [s, \phi_s])] \\
\qquad \rightarrow \\
\qquad (\leq^{\bigwedge}_{i\,(1,3)} [S] \rightarrow \leq^{\bigwedge}_{i\,(3,1)} [S]) \\
\quad] \\
\quad \rightarrow \\
\quad val_1(s, 1) \\
)
\end{array}
$$

This formula is of type Π_3; hence from the results presented in Table 2.2 it can be concluded that the encoding is adequate. Finally, note also that since EXISTS$^{\neg\emptyset}_{pref}$ and CRED$_{pref}$ can be reduced to EXISTS$^{\neg\emptyset}_{adm}$ and CRED$_{adm}$ respectively, the encodings given for the latter two decision problems in Propostion 3.3.3 give adequate encodings of EXISTS$^{\neg\emptyset}_{pref}$ and CRED$_{pref}$.

3.6 Encodings for the grounded valuation

A valuation v^3 is grounded for an ADF D if it is the least fixpoint of the characteristic operator for D. A natural choice for the defining formula for grounded valuations is by making use of the fact that this is equivalent to a valuation being minimal complete (Theorem 2.3.3). That a valuation v^3_g is a lower bound (with respect to $\leq_i$) for all complete valuations of an ADF D can be encoded via the following formula:

$$lb\text{-}comp_{g,h,j,k,l}[D] := \forall S_h^{\pm}[comp_{h,j,k,l}[D] \rightarrow \leq^{\bigwedge}_{i\,(g,h)} [S]]$$

as is expressed in the next lemma:

Lemma 3.6.1. Let $D = (S, C = \{\phi_s\}_{s \in S})$ be an ADF. Then

1. If $z^2 \models lb\text{-}comp_{g,h,j,k,l}[D]$ and z^2 coherent on $S_g^{\pm}$, then for v_g^3 such that $z^2 \;{}_{S_g^{\pm}}\!\rightleftarrows_S v_g^3$, it holds that v_g^3 is a lower bound of the set of all complete valuations for D.
2. If v_g^3 is a lower bound of the set of all complete valuations for D, then for any z^2 such that $z^2 \;{}_{S_g^{\pm}}\!\rightleftarrows_S v_g^3$, it holds that $z^2 \models lb\text{-}comp_{g,h,j,k,l}[D]$.

Proof. (sketch) Proof is similar to that of Lemma 3.3.4 using Propositions 3.1.1, 3.1.4, and Lemmas 3.3.2 and 3.4.2. □

That a valuation v_d^3 is grounded for an ADF D can then be encoded via the following formula:

$$ground_{d,e,f,g.h,j,k,l}[D] := comp_{d,e,f,g}[D] \wedge lb\text{-}comp_{d,h,j,k,l}[D]$$

as is stated in the next lemma:

Lemma 3.6.2. Let $D = (S, C = \{\phi_s\}_{s \in S})$ be an ADF. Then

1. If $z^2 \models \mathit{ground}_{d,e,f,g.h,j,k,l}[D]$, then z^2 is coherent on $S_d^\pm$ and for v_d^3 such that $z^2 \;{}_{S_d^\pm}\!\rightleftarrows_S\; v_d^3$, it holds that v_d^3 is the grounded valuation for D.
2. If v_d^3 is the grounded valuation for D, then for any z^2 such that $z^2 \;{}_{S_d^\pm}\!\rightleftarrows_S\; v_d^3$, it holds that $z^2 \models \mathit{ground}_{d,e,f,g.h,j,k,l}[D]$.

Proof. (sketch) Proof is similar to the proof of Lemma 3.3.5 using Lemma 3.6.2. □

Instantiating the defining formula for grounded valuations leads to a faithful encoding of ENUM$_{ground}$ into QENUM:

$$\mathit{ground}_{1,2,3,4.5,6,7,8}[D] := \mathit{comp}_{1,2,3,4}[D] \wedge \mathit{lb\text{-}comp}_{1,5,6,7,8}[D]$$

Nevertheless, making straightforward use of the defining formula we have given for grounded valuations for encoding the non-trivial existence problem as well as credulous and skeptical acceptance of statements (for grounded valuations) leads to formulas that are not adequate, reason for which a different approach is in order. One way of arriving at adequate encodings for these decision problems is by making use of a characterisation of grounded valuations given in [Strass and Wallner, 2013], for the statement of which we make use of the following definition:

Definition 3.6.1. Let $D = (S, C = \{\phi_s\}_{s \in S})$ be an ADF and v^3 be a valuation. The following are some properties v^3 can satisfy.

1. For each $s \in S$ such that $v^3(s) = 1$ there exists a $w^3 \in [v^3]_2$ and $w^3(\phi_s) = 1$.
2. For each $s \in S$ such that $v^3(s) = 0$ there exists a $w^3 \in [v^3]_2$ and $w^3(\phi_s) = 0$.
3. For each $s \in S$ such that $v^3(s) = \frac{1}{2}$ there exist $w_1^3 \in [v^3]_2$ and $w_2^3 \in [v^3]_2$ such that $w_1^3(\phi_s) = 1$ and $w_2^3(\phi_s) = 0$.

For convenience and reasons that will become clear soon we call any u^3 that satisfies properties one to three in Definition 3.6.1 for an ADF D a *candidate for the grounded valuation* of D.

Let

$$\mathcal{Y}_D := \{u^3 \mid u^3 \text{ is a candidate for the grounded valuation of } D\}$$

then the above mentioned characterisation of grounded valuations is expressed in the following Proposition [Strass and Wallner, 2013]:

Proposition 3.6.1. Let $D = (S, C = \{\phi_s\}_{s \in S})$ be an ADF. Then

$$\mathit{ground}(D) = \text{l.e. } \mathcal{Y}_D = \text{l.e. } \{u^3 \mid u^3 \text{ is a candidate for the grounded valuation of } D\}$$

An immediate corollary of Proposition 3.6.1 is:

Corollary 3.6.1. Let $D = (S, C = \{\phi_s\}_{s \in S})$ be an ADF, $s_* \in S$, and g^3 the grounded valuation of D. Then $g^3(s_*) = x$ if and only if $v^3(s_*) = x$ for every $v^3 \in \mathcal{Y}_D$ and $x \in \{0, 1\}$.

Proof. Assume first that $g^3(s_*) = x$ for $x \in \{0, 1\}$ and let $v^3 \in \mathcal{Y}_D$. Then from Proposition 3.6.1 it follows that $g^3 \leq_i v^3$ and, hence also $v^3(s_*) = x$ since 0 and 1 are the maximal elements of $\{1, 0, \frac{1}{2}\}$. The converse follows because also $g^3 \in \mathcal{Y}_D$. □

Encodings of EXISTS$^{\neg\emptyset}_{ground}$, CRED$_{ground}$ and SKEPT$_{ground}$ into QSAT can be provided based on Corollary 3.6.1. First, the following formulas encode the conditions that a valuation v^3_j has to satisfy in order for it to be a candidate for the grounded valuation for an ADF $D = (S, C = \{\phi_s\}_{s \in S})$:

- $g\text{-}cand^1_{j,k}[D](s) := val_j(s, 1) \rightarrow \exists S^{\pm}_{k^a_s}(ext_{k^a_s,j}[S] \wedge val_{k^a_s}(\phi_s, 1))$
- $g\text{-}cand^0_{j,k}[D](s) := val_j(s, 0) \rightarrow \exists S^{\pm}_{k^b_s}(ext_{k^b_s,j}[S] \wedge val_{k^b_s}(\phi_s, 0))$
- $g\text{-}cand^{\frac{1}{2}}_{j,k}[D](s) := val_j(s, \frac{1}{2}) \rightarrow \exists S^{\pm}_{k^c_s} \cup S^{\pm}_{k^d_s}(ext_{k^c_s,j}[S] \wedge val_{k^c_s}(\phi_s, 1) \wedge ext_{k^d_s,j}[S] \wedge val_{k^d_s}(\phi_s, 0))$

The next formula then encodes that a valuation v^3_j is a candidate for the grounded valuation for an ADF $D = (S, C)$:

$$g\text{-}cand_{j,k}[D] := coh_j[S] \wedge \textstyle\bigwedge_{s \in S}[g\text{-}cand^1_{j,k}[D](s) \wedge g\text{-}cand^0_{j,k}[D](s) \wedge g\text{-}cand^{\frac{1}{2}}_{j,k}[D](s)]$$

Stated formally:

Lemma 3.6.3. Let $D = (S, C = \{\phi_s\}_{s \in S})$ be an ADF. Then

1. If $z^2 \models g\text{-}cand_{j,k}[D]$, then z^2 is coherent on $S^{\pm}_j$ and for v^3_j such that $z^2 \;{}_{S^{\pm}_j}{\rightleftarrows}_S\; v^3_j$, it holds that v^3_j is a candidate for the grounded valuation for D.
2. If v^3_j is the grounded valuation for D, then for any z^2 such that $z^2 \;{}_{S^{\pm}_j}{\rightleftarrows}_S\; v^3_j$, it holds that $z^2 \models g\text{-}cand_{j,k}[D]$.

Proof. (sketch) Proof is similar to that of Lemma 3.2.1 making use of Propositions 3.1.1, 3.1.4, and Definition 3.6.1. □

The encodings for the various non-trivial (see Section 2.4) decision problems with respect to the grounded valuation we consider in this work can now be given based on Lemma 3.6.3, Corollary 3.6.1, as well as the fact that CRED$_{ground}$ and SKEPT$_{ground}$ are equivalent since there exists only one grounded valuation for any ADF:

Proposition 3.6.2. Given an arbitrary ADF $D = (S, C = \{\phi_s\}_{s \in S})$, let g^3 be the grounded valuation of D. Then the following hold:

1. ($\text{EXISTS}^{\neg\emptyset}_{ground}$) For some $s \in S$ it is the case that $g^3(s) \neq \frac{1}{2}$ if and only if $\forall S_1^{\pm}(g\text{-}cand_{1,2}[D] \rightarrow \bigvee_{s \in S} \neg val_1(s, \frac{1}{2}))$ is true.
2. (CRED_{ground} / SKEPT_{ground}) For an arbitrary $s_* \in S$, $g^3(s_*) = 1$ if and only if $\forall S_1^{\pm}(g\text{-}cand_{1,2}[D] \rightarrow val_1(s_*, 1))$ is true.

Proof. (sketch) Proof is similar to that of Proposition 3.2.1 using Proposition 3.1.3, Corollary 3.1.2, Lemma 3.6.3, Corollary 3.6.1, as well as the fact that CRED_{ground} and SKEPT_{ground} are equivalent. □

Proposition 3.6.2 gives faithful encodings of $\text{EXISTS}^{\neg\emptyset}_{ground}$, CRED_{ground} and SKEPT_{ground} into QSAT. The encodings are polynomial in the size of the ADF D. The following is a formula in prenex normal form equivalent to the encoding given in Proposition 3.6.2 for $\text{EXISTS}^{\neg\emptyset}_{ground}$ for an ADF $D = (S, C = \{\phi_s\}_{s \in S})$ where $S = \{s_1, s_2, ..., s_n\}$:

```
$\forall S_1^{\pm} S^{\pm}_{2^a_{s_1}} S^{\pm}_{2^b_{s_1}} S^{\pm}_{2^c_{s_1}} S^{\pm}_{2^d_{s_1}} ... S^{\pm}_{2^a_{s_n}} S^{\pm}_{2^b_{s_n}} S^{\pm}_{2^c_{s_n}} S^{\pm}_{2^d_{s_n}}$ (
    [
        $coh_1[S]$
            $\wedge$
        $\bigwedge_{s \in S}$(
            $(val_1(s,1) \rightarrow (ext_{2^a_s,1}[S] \wedge val_{2^a_s}(\phi_s, 1)))$
                $\wedge$
            $(val_1(s,0) \rightarrow (ext_{2^b_s,1}[S] \wedge val_{2^b_s}(\phi_s, 0)))$
                $\wedge$
            $(val_1(s,\frac{1}{2}) \rightarrow (ext_{2^c_s,1}[S] \wedge val_{2^c_s}(\phi_s, 1) \wedge ext_{2^d_s,1}[S] \wedge val_{2^d_s}(\phi_s, 0)))$
        )
    ]
        $\rightarrow$
    $\bigvee_{s \in S} \neg val_1(s, \frac{1}{2})$
)
```

The following formula in prenex normal form corresponds to the encoding for CRED_{ground} (and SKEPT_{ground}):

```
$\forall S_1^{\pm} S^{\pm}_{2^a_{s_1}} S^{\pm}_{2^b_{s_1}} S^{\pm}_{2^c_{s_1}} S^{\pm}_{2^d_{s_1}} ... S^{\pm}_{2^a_{s_n}} S^{\pm}_{2^b_{s_n}} S^{\pm}_{2^c_{s_n}} S^{\pm}_{2^d_{s_n}}$ (
    [
        $coh_1[S]$
            $\wedge$
        $\bigwedge_{s \in S}$(
            $(val_1(s,1) \rightarrow (ext_{2^a_s,1}[S] \wedge val_{2^a_s}(\phi_s, 1)))$
                $\wedge$
            $(val_1(s,0) \rightarrow (ext_{2^b_s,1}[S] \wedge val_{2^b_s}(\phi_s, 0)))$
                $\wedge$
            $(val_1(s,\frac{1}{2}) \rightarrow (ext_{2^c_s,1}[S] \wedge val_{2^c_s}(\phi_s, 1) \wedge ext_{2^d_s,1}[S] \wedge val_{2^d_s}(\phi_s, 0)))$
        )
```

$]$

$\rightarrow$

$val_1(s_*, 1)$

$)$

Both of the above formulas are of type Π_1; hence, based on the results presented in Table 2.2 it can be concluded that the encodings are adequate.

CHAPTER 4

Implementation

In this chapter we carry out first steps in direction of determining the possible benefits of implementing a QBF-based reasoning system for ADFs based on the encodings we have presented in Chapter 3. Specifically, in Section 4.3 we describe a prototype of such a system that we have implemented. The purpose of Section 4.4 is then to set a first benchmark for QBF based implementations of reasoning on ADFs by reporting on preliminary experiments determining the performance of the prototype system. We also compare its performance with that of `DIAMOND`, the only other available reasoner for ADFs with respect to the semantics we consider in this work that is, to the best of our knowledge, currently available.

Section 4.1 puts the work we present in this chapter in context by very briefly surveying implementation methods for abstract argumentation, also covering software systems available for ADFs. In Section 4.2 we survey the main approaches existing to date for implementing QSAT-solvers, also introducing the QSAT-solver `DepQBF` that we use as back-end for the prototype system we describe in this chapter.

4.1 Overview of implementation methods and systems for abstract argumentation

An implementation method for abstract argumentation can be considered to be any method which enables solving the various reasoning tasks (see Section 2.4) defined on abstract argumentation frameworks computationally. To date, several different approaches for developing such methods exist, especially for Dung style frameworks. There are also various software systems which implement some of these methods, although an exhaustive empirical evaluation of the latter has yet to be carried out.

An in-depth and general overview of research into implementation methods for argumentation is out of the scope of this work, so in this section we only point out some of the main strategies that have been developed for Dung argumentation frameworks. These have, naturally, also been those argumentation frameworks which have received the most attention to date. We focus in a little more detail on implementation methods that are based on formal logic and also provide some insight into the main methods and software systems developed to date for abstract dialectical frameworks.

As has already been indicated in the introduction to this work, a recent survey of the approaches and advances in implementation methods for argumentation can be found in [Charwat et al., 2013]. The authors of this work identify two general requirements that are crucial for the applicability of such methods. In the first place, implementation methods for abstract argumentation must have a certain level of generality so that the various semantics that are defined for them can be treated; and in the second place they must also have a sufficient level of efficiency to deal with the inherent complexity of many of the reasoning problems defined for argumentation.

The above mentioned technical report also usefully classifies the two main approches towards the development of implementation methods for abstract argumentation into reduction based and direct approaches. The first aim to translate the reasoning problems defined on argumentation frameworks to some other formalism for which efficient systems exist, while direct approaches are tailored specifically to reasoning on abstract argumentation systems.

An important instance of direct approaches for the development of implementation methods is in the first place the labelling approach in which the idea is to enumerate the labellings of an AF using various techniques to prune the space of possible labellings that has to be explored. Instances of the labelling approach are presented in [Doutre and Mengin, 2001, Modgil and Caminada, 2009, Nofal et al., 2012, Verheij, 2007] and implementations of this approach are the `PyAAL` [1] [Podlaszewski et al., 2011] and `COMPARG` [2] systems. Other direct approaches are based on characterising the acceptance status of arguments with respect to an AF and semantics in terms of winning strategies in certain dialogue games on the AF. Algorithms based on this idea are presented in [Modgil and Caminada, 2009, Thang et al., 2009], `Dungine` [3] [South et al., 2008] and `Dung-O-Matic` [4] being software systems based on this approach. Finally, dynamic programming approaches use insights from fixed parameter tractability theory to operate on decompositions of argumentation frameworks [Dvořák et al., 2012a]. `dynPARTIX` [5] [Dvorák et al., 2011] is a system implementing this last approach.

Reduction based approaches for implementing abstract argumentation can, themselves, be classified according to the target formalism into which the reasoning problems defined for argumentation systems are translated, some of the main ones being propositional logic, answer set programming, and constraint satisfaction problems. Reductions to constraint satisfiaction problems are presented in [Amgoud and Devred, 2011, Bistarelli and Santini, 2011] and are implemented in the system `ConArg` [6]. Other options for target formalisms for reducing reasoning on AFs that have been explored in the literature are sets of equations [Gabbay, 2012a, Gabbay, 2012b] and monadic second order logic [Dvořák et al., 2012b].

[Besnard and Doutre, 2004] seems to be the first work that advocates the use of propositional logic for evaluating AFs. The authors of this work use the extensional characterisation of

[1]`http://heen.webfactional.com/`

[2]`http://www.ai.rug.nl/~verheij/comparg/`

[3]`http://www.argkit.org/`

[4]`http://www.arg.dundee.ac.uk/?page_id=279`

[5]`http://www.dbai.tuwien.ac.at/proj/argumentation/dynpartix/`

[6]`http://www.dmi.unipg.it/francesco.santini/argumentation/conarg.zip`

the semantics for AFs to give encodings into SAT for the verification problem for admissible, complete and stable extensions. They also present encodings of the problems of enumerating the stable, admissible, preferred, complete, and grounded extensions of a given AF into the problem of enumerating different types of models of a propositional formula.

As an example, for a given AF $F = (A, R)$, there is a one to one correspondence between the models of the formula

$$adm[F] := \bigwedge_{a \in A}((a \rightarrow \bigwedge_{b \in par_R(a)} \neg b) \wedge (a \rightarrow \bigwedge_{b \in par_R(a)}(\bigvee_{c \in par_R(b)} c)))$$

when projected onto the variables in A and the admissible extensions of F. The preferred extensions of F can now be characterized as those extensions associated to any subset-maximal model of $adm[F]$ via this correspondence. A different translation of evaluation of AFs with respect to the different semantics is carried out in [Gabbay, 2011] via the Peirce-Quine dagger connective $\downarrow$, which is defined as $a \downarrow b \equiv \neg a \wedge \neg b$.

Quantified boolean logic is the target formalism chosen in [Arieli and Caminada, 2012,Arieli and Caminada, 2013] for translating the reasoning problems associated to AFs and, therefore, serves as direct inspiration for our work. Specifically, in this work the labelling based definition of most of the major semantics defined for AFs are used as a basis for reductions of the enumeration problems for these semantics into the model enumeration problem for quantified boolean logic. Since three labels are considered for the definition of the semantics ($\{\text{true}, \text{false}, \text{undefined}\}$), signed variables are used to encode the different labels that can be assigned to a given argument in an AF and attention is restricted to coherent valuations in very much the same manner as we have done in Section 3.1 in this work. The possibility to express maximisation and minimsation directly in quantified boolean logic (versus propositional logic where this is not possible) plays an important role in the encodings given in [Arieli and Caminada, 2013] for semi-stable, eager, stage, grounded and preferred extensions, the latter of which, given the similarity between the definitions of this semantics for AFs and ADFs, mirrors the defining formula of preferred valuations of ADFs presented in Section 3.5.

Contrasting with the "monolithic" reduction methods based on propositional (and quantified boolean logic) presented until now is the iterative SAT based procedure introduced in [Dvořák et al., 2012] which is especially well suited for deciding credulous and skeptical acceptance of arguments in AFs for those semantics for which these problems are complete for the second level of the polynomial hierarchy (i.e. one call to a SAT solver will not suffice in general). The basic idea underlying this approach is to iteratively modify propositional queries about some "less complex" semantics from which the extensions with respect to the semantic of interest can be derived based on the outcome of these queries when posed to a SAT-solver.

In the case of "preferred" semantics, for example, this procedure amounts to simulating, via repeated calls of a SAT-solver with queries selecting some subset of the models corresponding to admissible extensions, the search for an example (in the case of credulous acceptance) or a counterexample (in the case of skeptical acceptance) of a maximal admissible extension containing, respectively not containing the statement whose acceptance is under scrutiny. This procedure has been implemented in the form of the system `CEGARTIX`[7] for acceptance of arguments under preferred, semi-stable and stage semantics. A more recent example of an iterative

[7]`http://www.dbai.tuwien.ac.at/proj/argumentation/cegartix/`

SAT based approach is that presented in [Cerutti et al., 2013], which is tailored to computing the preferred labellings of AFs. Finally, in [Wallner et al., 2013] algorithms based on extensions of SAT (computing a backbone or minimal correction sets) are used to implement computing extensions with respect to semi-stable, ideal, and eager semantics for AFs.

An alternative choice of target formalism for reduction based implentations of abstract argumentation is the so called "answer set programming" (ASP) paradigm [Brewka et al., 2011], itself a declarative fragment of logic programming with ever more efficient solvers available (see, for example, [Gebser et al., 2011, Leone et al., 2006]). A basic kind of logic program, called a "normal logic program", is a finite set of rules of the form

$$a \text{ :- } b_1, ..., b_m, \text{not } c_1, ..., \text{not } c_n$$

where ":-" can be likened to the implication symbol of propositional logic (written from right to left) $\leftarrow$ and $a, b_1, ..., b_m, c_1, ..., c_n$ are "atoms", i.e. expressions of the form $p(t_1, ..., t_n)$ with p being a "predicate name", each t_i $(1 \leq i \leq n)$ being either a variable or a constant from a fixed domain of elements. What is at the right of the symbol ":-" in a rule is called the "body" while that at the left is the "head". not stands for "negation as failure" or "default negation", and "not a" for some atom a can intuitively be understood as being true unless a is proven.

Answer sets are one of the most prominent approaches devised to define the semantics of logic programs with default negation and are defined as the subset minimal models of the so called "Gelfond-Lifschitz reduct" of the program [Gelfond and Lifschitz, 1988]. Deciding whether a normal propositional (i.e. without variables) program has an answer set is an NP-complete problem, while higher expressivity is achieved by extending normal logic programs through further language constructs like disjunction, weight constraints or aggregates [Simons et al., 2002, Leone et al., 2006].

The most common approach to use answer set programming to evaluate abstract argumentation frameworks with respect to the different semanticsis by providing a logic program $\pi_1 \cup \pi_2$ where π_1 are the rules that represent the evaluation of a given framework with respect to the semantics in question and π_2 is a set of rules with empty bodies (the "facts") that encode the framework. The answer sets then represent the extensions, labellings or valuations of the AF or ADF.

A relatively recent survey on the use of answer set programming for the evaluation of Dung style argumentation frameworks is [Toni and Sergot, 2011] and [Nieves et al., 2008, Wakaki and Nitta, 2008, Egly et al., 2010] are some of the most important instances of this approach. Notably, the last of these is also at the base of the `ASPARTIX` system [8] which, apart from the semantics for Dung style argumentation frameworks, also implements encodings for preference based, value based and bipolar frameworks.

Particularly significant for the present work is that the first systems available for reasoning on abstract dialectical frameworks are also based on the answer set programming paradigm. The first of these is `ADFSys` [9] [Ellmauthaler and Wallner, 2012, Ellmauthaler, 2012] which computes the admissible, model, stable, preferred and well-founded valuations of an ADF as defined in

[8] `http://www.dbai.tuwien.ac.at/proj/argumentation/systempage/`

[9] `http://www.dbai.tuwien.ac.at/proj/argumentation/adfsys/`

the initial work on ADFs [Brewka and Woltran, 2010] based on the propositional representation of ADFs.

In ADFSys admissible, model and stable semantics are encoded via normal logic programs, while well-founded and preferred semantics are encoded using disjunctive ASP and optimization programs respectively for which the problem of deciding whether a program has an answer set is in Σ_2^P in general. The first allow for disjunction in the head of the rules of a logic program, while the second include special aggregates which allow to specify subset minimization. Since some of the semantics as defined in [Brewka and Woltran, 2010] require a distinction between different link-types of the ADFs, ADFSys also requires that for some of the semantics the link-type be given as input to the program or otherwise these are determined automatically using the so called "saturation technique" [Eiter and Gottlob, 1995] which again is achieved using disjunctive ASP.

DIAMOND [Ellmauthaler and Strass, 2013] is a more recent system for evaluating ADFs which extends the ASP based approach of ADFSys to the new semantics of ADFs as given in [Brewka et al., 2013] and builds on the Potsdam Answer Set Solving Collection (Potassco) [Gebser et al., 2011]. In this case the functional representation of the acceptance conditions is used, making it possible to implement evaluation of ADFs even with respect to complete and admissible semantics using normal logic programs via a module which implements the characteristic operator for ADFs. In any case, preferred semantics still requires the use of subset minimization which is implemented via the aggregate for minimization that is also used in ADFSys. A program that transforms acceptance functions given as propositional formulas into the functional representation used by DIAMOND completes the software bundle for ADF based argumentation available at the website dedicated to this system [10].

4.2 Overview of QSAT solving strategies

In the following we give a brief overview of the main approaches that exist to current date for the development of solvers dedicated to the satisfiability problem of QBFs as well as pointers to literature on current existing systems. We also introduce the main ideas behind the QSAT solver used as a reasoner for the system we present here. This section is based on [Woltran, 2003, Kroening and Strichman, 2008, Giunchiglia et al., 2009, Bubeck, 2010, Lonsing, 2012].

Most approaches developed to date for solving the satisfiability problem of QBFs assume the input formula to be in PCNF, therefore we also focus on these types of formulas here. Also, most approaches are, quite naturally, motivated by existing approaches in the very well developed area of (propositional) SAT-solving. Thus, work on this topic got started with the work of [Büning et al., 1995] where the resolution procedure of propositional logic [Robinson, 1965] is generalised to the case of quantified boolean logic. This generalisation is referred to as "Q-resolution" and is defined in terms of the notion of a "Q-resolvent" of two clauses occurring in a QBF in PCNF:

[10] https://isysrv.informatik.uni-leipzig.de/diamond

Definition 4.2.1. Let ϕ be a QBF in PCNF with prefix $\mathcal{Q}$, matrix ψ and c_1 and c_2 two clauses in the matrix with c_1 containing an existential literal l and c_2 containing the literal $\bar{l}$. Then the *Q-resolvent* is obtained from c_1 and c_2 as follows:

1. Eliminate all occurrences of l and $\bar{l}$ from $c' := c_1 \vee c_2$.
2. If c' is a tautological clause no resolvent exists. Otherwise, the Q-resolvent is obtained from c' by removing the universal literals whose level is lower than the level of all the existential literals in c'.

A *resolution proof* from a QBF in PCNF ϕ is a sequence of clauses that are either in the matrix of ϕ or obtained via resolution from clauses preceeding them in the sequence. Now, although naive search for a Q-resolution proof is computationally expensive, it holds that a QBF in PCNF is false if and only if there exists such a resolution proof ending with a non tautological clause containing only universal literals. It has also been shown that in some limited scenarios (mainly evaluating to false) resolution based QSAT solvers can achieve good results compared to state of the art solvers [Giunchiglia et al., 2001a, Giunchiglia et al., 2001b].

Most existing succesful QSAT solvers to date are based on so called "search" or, alternatively, "variable elimination" approaches. Search based procedures are extensions of the DPLL method [Davis et al., 1962] which is at the base of most current SAT solvers for propositional logic. This method follows closely the semantics of QBFs to determine the satisfiability of a formula.

The following is a formulation of the QDPLL procedure as presented in [Giunchiglia et al., 2009] based on the specification of this procedure in [Cadoli et al., 1998], which is the first work to present a QBF-engine based on the DPLL method.

```
0 function QDPLL($\phi, \mu$):
1   if a contradictory clause is in the matrix of $\phi_\mu$ return FALSE
2   if the matrix of $\phi_\mu$ is empty return TRUE
3   if $l$ is unit in $\phi_\mu$ return QDPLL($\phi, \mu; l$)
4   if $l$ is monotone in $\phi_\mu$ return QDPLL($\phi, \mu; l$)
5   $l$ := a literal at the highest level in $\phi_\mu$
6   if $l$ is existential return QDPLL($\phi, \mu; l$) or QDPLL($\phi, \mu; \bar{l}$)
7   else return QDPLL($\phi, \mu; l$) and QDPLL($\phi, \mu; \bar{l}$)
```

QDPLL's input is a QBF ϕ in PCNF and a sequence of literals μ. Initially, ϕ is the formula whose satisfiability is under scrutiny and μ is the empty sequence. It is assumed that ϕ does not contain any tautological clauses (see Lemma 2.2.2). We remind the reader that ϕ_μ denotes propagation on the sequence of literals μ (see Definition 2.2.8).

The core of the algorithm is specified in lines 5-7. Essentially, QDPLL determines the satisfiability of ϕ following the recursive definition of the semantics of a QBF as defined in Definition 2.2.5. Since $\phi = Q_1p_1Q_2p_2...Q_np_n\psi$ is in PCNF this boils down to following the prefix of the formula from left to right considering, for each occurence Q_ip_i ($1 \leq i \leq n$) in the prenex for $Q \in \{\exists, \forall\}$, whether the conditions imposed by the semantics for the formula to be true hold. In the first call to QDPLL this means checking that $Q_2p_2Q_3p_3...Q_np_n\psi[p_i/\bot]$ or $Q_2p_2Q_3p_3...Q_np_n\psi[p_i/\top]$ are true if $Q_1 = \exists$. In the case of $Q_1 = \forall$, both of the last formulas must be shown to be true. This is achieved by calling QDPLL with the result of propagating p_1 or (/and) $\neg p_1$ in ϕ (see Definition 2.2.8). The same process is repeated for successive calls to Q-DPPL with μ keeping track of the decisions taken when branching out on $\bot$ or $\top$ for variables occuring in the prenex of ϕ.

The procedure terminates if this repeated process of branching on $\bot$ or $\top$ and propagating accordingly leads to an empty clause (the matrix of ϕ is not satisfied by μ) or an empty set of clauses (the matrix of ϕ is satisfied by μ) in the matrix of ϕ. These two possibilities are covered by lines 1 and 2 (the empty clause being a special case of a contradictory clause). Finally, lines 1,3, and 4 of QDPLL are optimizations based on the observations in Lemma 2.2.2. Also, note that branching out on variables (line 5 in QDPLL) is done according to the order imposed by the level in which the variables occur in ϕ. The fact that *any* variable occurring in the same level can be considered when branching is due to the fact that quantifiers occurring in the same quantifier block can be re-ordered while preserving semantical equivalence.

Just as is the case for SAT solvers and DPLL, more recent QSAT solvers differ considerably from the formulation of QDPLL as given above. One important development is the extension of clause and cube learning approaches in SAT-solving to the QSAT scenario. Clause learning approaches, for example, use Q-resolution to add clauses derived from the original formula to the formula under consideration, aiming at guiding the search process out of regions of the search space which do not contain solutions [Lonsing, 2012]. Approaches of this sort are presented in [Giunchiglia et al., 2002,Letz, 2002,Zhang and Malik, 2002a,Zhang and Malik, 2002b]. Another natural optimization of QDPLL as presented above is to incorporate parallelism. Approaches of this sort are presented in [Feldmann et al., 2000, Lewis et al., 2009, Klieber et al., 2010, Lewis et al., 2011].

Non determinism in Q-DPPL as presented above appears in step 5 (as well as when deciding to branch out on QDPLL($\phi, \mu; l$) or QDPLL($\phi, \mu; \bar{l}$) in lines 6 and 7), the order in which variables are chosen in this step potentially having a great impact on the performance of a QSAT-solver. According to [Lonsing, 2012] there is to date no comprehensive empirical study of the effect different branching heuristics may have. On the other hand, [Rintanen, 1999b] presents ideas based on the inversion of quantifiers to overcome the limitation of QDPLL having to branch on literals at the highest level. Finally, QDPLL as well as its extensions have been generalised to QBFs not in PCNF in [Egly et al., 2006, Egly et al., 2009, Goultiaeva et al., 2009, Goultiaeva and Bacchus, 2010].

The other main family of QSAT-solvers follow what has come to be called the "variable elimination" approach [Giunchiglia et al., 2009]. The main characteristic of this approach is that, in order to determine whether a closed QBF ϕ is true or not, ϕ is repeatedly transformed into a semantically equivalent formula ψ where some variable (as well as corresponding quantifier) in ϕ has been eliminated. This procedure continues until the resulting formula can be easily determined to be true or false. Usually this “variable elimination” procedure increases the size of the original formula and, hence, the success of algorithms based on this approach is determined by the amount of that size increase. An important difference between search based and variable elimination based algorithms in the case of QBFs in PCNF is that, while the first follow the semantics of QBFs considering variables in the prefix from left to right, the latter usually consider the variables for elimination from right to left.

The most straightforward variable elimination algorithm for formulas in PCNF repeatedly applies the equivalences $\exists p\phi \equiv \phi[p/\top] \vee \phi[p/\bot]$ and $\forall p\phi \equiv \phi[p/\top] \wedge \phi[p/\bot]$ to quantified subformulas of the original formula, i.e a formula ϕ of the form

$$Q_1p_1Q_2p_2...Q_np_n\psi$$

with $Q_i \in \{\forall, \exists\}$ for $1 \leq i \leq n$ is *expanded* into the equivalent QBF ϕ'

$$Q_1p_1Q_2p_2...Q_{n-1}p_{n-1}(\psi[p/\top] \otimes \psi[p/\bot])$$

where $\otimes$ is $\wedge$ if $Q_n = \forall$ and $\vee$ if $Q_n = \exists$. In theory this procedure can be continued until all variables are eliminated and a SAT solver could then be used on the remaining formula to determine the satisfiability of ϕ. In practice this is unfeasible because the number of clauses in the resulting formula will be $m * 2^n$ where m is the number of clauses in the initial formula ϕ but various optimisations and generalisations are available which could potentially make this straightforward approach more efficient. Also, expansion in accordance with the equivalences $\exists p\phi \equiv \phi[p/\top] \vee \phi[p/\bot]$ and $\forall p\phi \equiv \phi[p/\top] \wedge \phi[p/\bot]$ can be applied to non PCNF QBFs, leading to non PCNF solvers. Also data structures which allow for more compact representations of QBFs can be useful for carrying out these expansions in a more efficient manner [Ayari and Basin, 2002, Bubeck and Büning, 2007, Lonsing and Biere, 2008, Pigorsch and Scholl, 2009, Reimer et al., 2011, Janota et al., 2012].

A different technique is to generalise the Davis Putnam (DP) algorithm for the SAT problem [Davis and Putnam, 1960] to QSAT by eliminating existential variables in a QBF ϕ in PCNF via resolution. The basic idea is to consider variables from right to left as they appear in the prefix of ϕ, eliminating existentially quantified variables by resolving all clauses in the matrix of ϕ in which the variable appears positively with all clauses in which the variable appears negatively. Variables that are quantified universally can then be eliminated via universal reduction (see Lemma 2.2.2). Further optimisations are checking for contradictory clauses, and propagating unit and montone literals as in QDPLL. An algorithm along these lines based on the presentation in [Giunchiglia et al., 2009] is the following:

```
0 function QDP($\phi$):
1   if a contradictory clause is in the matrix of $\phi$ return FALSE
2   if the matrix of $\phi$ is empty return TRUE
3   if $l$ is unit in $\phi$ return QDP($\phi_l$)
```

```
4    **if** $l$ is monotone in $\phi$ **return** QDP($\phi_l$)
5    $z$ := a variable at level 1 in $\phi$
6    **if** $z$ is existential **return** QDP($resolve(z, \phi)$)
7    **else return** QDP($universal - reduction(z, \phi)$)
```

One of the main drawbacks of QDP as specified above is the exponential explosion of the size of the input formula that results from resolution. In the worst case, when eliminating a existential variable p in a formula ϕ, half of the clauses in the formula contain p and the other half $\neg p$ in which case $m^2 - m$ clauses remain after resolving on p. Therefore, $\mathcal{O}(m^{2^n})$ clauses are generated after resolving n times.

But since there cannot be more than 3^n distinct clauses (where n is the number of variables in the original formula), the above is only true if the creation of duplicate clauses is not prevented. In fact, other redundancies in the clauses generated by resolution can occur and hence approaches like subsumption removal [Biere, 2004, Zhang, 2005] or methods of preprocessing [Biere et al., 2011, Eén and Biere, 2005, Giunchiglia et al., 2010b] can be applied to reduce the size of the formula further.

While "search based" and "variable elimination" based QSAT solvers are the main types of QSAT solvers in existence to date other approaches exist. In particular, it is possible to combine both approaches as has been proposed in different ways in, for example, [Benedetti, 2005a, Pulina and Tacchella, 2007]. Alternatively, QSAT can be encoded into a different formalism as has been done in [Donini et al., 2002] where QBFs are encoded in the language of the model checker NuSMV.

Finally, other developments worth mentioning are the use of binary decision diagrams for QBF solving in search based approaches [Audemard and Sais, 2005] as well as variable elimination approaches [Pan and Vardi, 2004, Jussila et al., 2006, Olivo and Emerson, 2011] and the use of the technique of skolemisation to eliminate existential variables from a QBF [Benedetti, 2005b, Jussila et al., 2007].

We conclude this section by briefly introducing `DepQBF` [11] [Lonsing and Biere, 2010a], which is the QSAT solver we use as back-end of the prototype software sytem we present in Section 4.3. `DepQBF` is a search based QSAT solver for QBFs in PCNF based on QDPLL and incorporating clause and cube learning. The main feature of `DepQBF` is the integration of so called "dependency schemes" into QDPLL [Samer and Szeider, 2009]. Informally, dependency schemes are binary relations on the variables ocurring in a QBF that capture "dependency" among variables in the sense that if D is a dependence scheme and x and y two variables ocurring in a QBF such that $(x, y) \in D$, then the result obtained from choosing y before x in step 5 of QDPLL as defined in the previous section may not be sound.

QDPLL as defined above is based on the dependency scheme induced on the variables occurring in a QBF by the level in which they appear as defined in Section 2.2. This is reflected by the fact that variables are selected in step 5 of QDPLL according to their level (from highest to

[11] `http://lonsing.github.io/depqbf/`

lowest). The use of a less restrictive dependency scheme (a dependency scheme D is less restrictive than a dependency scheme D' if and only if $|D| \subseteq |D'|$) as is the case of `DepQBF` allows more freedom in the search of assignments that make a QBF satisfiable. Moreover, in [Lonsing and Biere, 2010b] the notion of a unit literal as well as clause and cube learning are generalised to arbitrary dependency schemes and these generalisations are incorporotad into `DepQBF`. Other optimisations to QDPLL implemented in `DepQBF` such as efficient detection of unit and pure literals, removal of learnt constraints and restarts are described in [Lonsing and Biere, 2010a].

Experiments presented in [Lonsing and Biere, 2010b] show that QDPLL integrated with the so called "standard dependency scheme" outperforms QDPLL without dependency schemes based on the prefix of a QBF as well as other dependency schemes such as those based on quantifier trees which are more restrictive. For this reason, `DepQBF` uses this dependency scheme to boost QDPLL as its default choice. With this setting, `DepQBF` was placed first in the main track of QBFEVAL 2010, the main competition for QSAT solvers, in a score-based ranking.

As a final note about QSAT solvers, it should be observed that a recent trend [Giunchiglia et al., 2010b, Lonsing and Biere, 2011, Gelder et al., 2012] in the development of QSAT technology is paying attention to pre-processing techniques on the input to QSAT solvers, which can lead to an impressive reduction in formula size and solving time in practice. For the prototype software system we present in this chapter, in particular, we use `Bloqqer` [12] [Biere et al., 2011].

4.3 Description of a prototype system for reasoning on ADFs via QBFs

In the following we describe the prototype system we have implemented, `QADF` [13], that enables reasoning on ADFs based on the encodings in quantified boolean logic presented in Section 3. `QADF` itself can be seen to be somewhat analogous to a compiler which, when given a representation of an ADF and a reasoning task of choice, outputs the encoding of the reasoning task as a QBF. When combined with a QSAT solver of choice, the result is that one has a system implementing the various reasoning tasks we have considered in this work for ADFs.

At the webpage where `QADF` is available (see Footnote 13) we also provide a Linux shell script `QADF2Dep` that uses the QSAT solver `DepQBF` described in the previous section as a backend for a working reasoner for ADFs. Crucial is also that the pre-processing tool `Bloqqer` is used to transform the output of `QADF` before feeding it to `DepQBF`. More information on running `QADF` as well as `QADF2Dep` is also available on this webpage.

`QADF` is implemented in the statically typed multi-paradigm programming language Scala [Odersky et al., 2011], itself implemented in Java. Thus, `QADF` runs as a Java application, making it possible to run it on any system where the Java runtime environment is installed. Scala is especially well suited for writing applications combining the object oriented and functional programming paradigms. True to the "Scala programming philosophy" in our implementation

[12] `http://fmv.jku.at/bloqqer/`
[13] `http://www.dbai.tuwien.ac.at/proj/adf/qadf`

we have also refrained from using constructs of the language that have the potential of generating side-effects in so far as is reasonable.

Use of the program

`QADF` is called via command line using the following commands:

```
qadf [options] semantics inputfile
```

The parameter "semantics" is required to be one of "-3m" (three valued models), "-2m" (two valued models), "-adm" (admissible), "-comp" (complete), "-pref" (preferred), "-ground" (grounded) representing the various semantics defined for ADFs. The optional parameters are as follows:

-h (print usage information)

-E (generate encoding for existence problem)

-N (generate encoding for non trivial existence problem)

-C s (generate encoding for credulous acceptance of the statement s)

-S s (generate encoding for skeptical acceptance of the statement s)

-o outputfile (output encoding to "outputfile")

-m (print mapping from integers used as (signed) propositional variables in encoding to statements they represent in the input ADF)

When called without any of the options for the encodings, `QADF` generates the encoding for the enumeration problem. When called without the option "-o outputfile", `QADF` prints the encoding to standard output.

Input: representation of ADFs

For compatibility, for the representation of the ADFs that are given as input to `QADF` the propositional representation that is also used in the systems `ADFSys` and `DIAMOND` is used. The input file should contain an expression "statement(s)." and an expression "ac(s,acceptance-condition)." (note the period at the end of these expressions) for each statement in the ADF, where "s" is the name of the statement and "acceptance-condition" is the acceptance condition associated to statement with identifier "s".

Acceptance conditions, in turn, are constructed out of any of the names of the statements in the ADF, the primitives "c(v)" denoting $\top$ and "c(f)" denoting $\bot$ as well as unary and binary constructors representing the various propositional conectives. This must be in accordance with the recursive definition of propositional formulas as given in Definition 2.1.2. The symbols for the connectives used in the input file are "neg" for $\neg$, "and" for $\wedge$, "or" for $\vee$, "imp" for $\rightarrow$ and "iff" for $\leftrightarrow$.

Example 4.3.1. The following is the ADF from Example 2.3.1 in the input format of `QADF`:

```
statement(a).
statement(b).
statement(c).
statement(d).
ac(a,a).
ac(b,b).
ac(c,and(a,neg(b))).
ac(d,and(neg(a),b)).
```

Output: Q-DIMACS format

As has already been indicated, `QADF` outputs the encodings in the Q-DIMACS format, which is the input format that most currently available QSAT solvers assume. Q-DIMACS is itself an extension of the DIMACS format for representing propositional formulas in CNF that is assumed by most SAT solvers in existence to date. Q-DIMACS basically extends DIMACS by adding lines for each quantifier block in the prefix before the usual DIMACS representation of the matrix of the QBF being represented.

General structure of the program

We have organised the source code of the project into several packages which group together related classes:

- main
- parser
- bf
- qbf
- adf
- functions
- encodings
- util

"main" contains the "Main" class (providing entry to the program) and "util" contains miscellaneous auxilliary classes. The content of the other packages are described in a little more detail in the following.

parser

The package "parser" contains the parser of the input file in the format described in Section 4.3 representing the ADF for which an encoding is to be constructed by `QADF`. For parsing we used the "parser combinators" provided by Scala, a library of functions and operators that serve as building blocks for parsers and have a one to one correspondence to the constructions of a context free grammar [Odersky et al., 2011].

bf / qbf

These packages contain classes representing propositional (bf) and quantified boolean formulas (qbf). They are implemented as Scala "case classes" which allow for easy pattern matching. This enables to define methods and functions on propositional and quantified boolean formulas in a recursive manner very much like that which is usual in formal logic.

adf

This pacakge contains a class representing ADFs. ADFs are implemented in a manner very much reflecting their definition, i.e. as a pair consisting of a Scala Set of boolean or propositional variables representing the set of statements of an ADF and a Scala Map between propositional variables and propositional formulas representing the set of acceptance conditions yet allowing for efficient data access.

functions

The "functions" package contains a singleton object containing various auxilliary functions useful for the encodings as the functions τ_1, τ_2, and val_j defined in Section 3.1.

encodings

The "encodings" package is obviously the crucial package of the program and contains all the classes corresponding to the encodings. The various classes corresponding to the different encodings themselves inherit from general classes for encodings.

The main implementation issue to generate the encodings has been to strike a balance between modularity and a reasonable degree of efficiency for a prototype system. For the latter reason the most straightforward approach to implement the encodings which would be to generate the whole encoding in non PCNF form in memory and do wholesale transformation of the encodings to (PCNF and) Q-DIMACS has not been the one followed in the implementation we present. Rather, the encodings are generated more or less directly in PCNF but in a modular form and transformation to the Q-DIMACS format is also done in a modular fashion. In particular, transformation functions (for CNF conversion) are only called in the program for conversion of the acceptance formulas. Slight technical complications then arise because of the need to deal with labels introduced by the Tseitin procedure described briefly in Sections 2.1 and 2.2 for the conversion to PCNF as well as the fact that the Q-DIMACS format requires a header (appearing before the actual formula) stating number of variables and clauses.

The use of the Tseitin procedure for transformation into PCNF means the encodings generated by `QADF` do not correspond directly to those presented in Chapter 3. Also, although the Tseitin procedure in principle requires introducing labels for every subformula of the encoding, our implementation is somewhat optimised in order to generate as few labels as possible while keeping the encodings polynomial in the size of the input ADF; in particular, labels are introduced for all subformulas only in the case of the acceptance formulas that appear in the encodings. Finally, "jumping" a level in the polynomial hierarchy (in cases in which, as is the case of the encoding corresponding to CRED$_{adm}$, the innermost quantifier block of the original encoding is quantified universally) because the labels generated by the Tseitin procedure need to appear quantified existentially in the innermost block of the resulting encoding is avoided by generating instead a formula that is unsatisfiable if and only if the original encoding is satisfiable.

4.4 Experiments

In this section we finally report on preliminary experiments carried out in order to set a first benchmark for the performance of QBF-based reasoning systems based on the encodings presented in this work and as implemented by `QADF`. We first desribe the experiment set-up and then proceed to present the results as well as some first conclusions regarding performance that can be reached based on these experiments.

Experiment setup

Our preliminary experiments focus on the reasoning tasks CRED$_{adm}$ and SKEPT$_{pref}$, which are somewhat representative of the different reasoning tasks for ADFs for which we provide encodings in this work. For comparative purposes, we have also determined the performance of our system relative to that of `DIAMOND` described in Section 4.1.

All experiments were carried out on an `openSuse` (11.4) machine with eight `Intel Xeon` processors (2.33 GHz) and 48 GB of memory. Apart from `QADF` (version 0.1) we use the latest version of `DIAMOND` (version 0.9) which, in turn, requires `gringo` (version 3.0.4), `clasp` (version 2.1.5), `claspD` (version 1.1.4) from the `Potassco` bundle of answer set programming tools. `DIAMOND` only does valuation enumeration out of the box so we adapted it (using ASP constraints) in order to carry out credulous and skeptical reasoning. For the ADF reasoning system based on the QBF encodings presented in this work we use the pre-processing tool `Bloqqer` (version 031) and the QSAT solver `DepQBF` (version 2.0) described in Section 4.2. For simplicity we will call the whole system (i.e. `QADF` + `Bloqqer` + `DepQBF`) just `QADF` in the following.

In order to generate random instances of ADFs for our experiments we have used the generator used to test `ADFSys` [Ellmauthaler, 2012], which in turn is based on a generator for AFs used to determine the performance of `ASPARTIX` and `CEGARTIX` (see Section 4.1). The basic idea behind this generator is that a predetermined number of statements are placed on a grid with directed edges and a certain width, an outgoing edge from one neighbor to the other indicating that the one neighbor has a certain probability to appear in the acceptance condition of the other. Apart from this aspect, there are probabilities assigned to the links in the grid governing whether both statements in a link appear in the acceptance condition of each other ("symmetric relation of attack or support"), whether the statement is mutated into one of $c(v)$ or $c(f)$, and whether each of the statements (or constant in case the statement has been mutated) in question will appear negated or not in the relevant acceptance condition. Finally, for constructing the acceptance conditions there is a probability that determines whether the connective appearing between the part of the acceptance condition that has already been constructed and a new statement that is added to the condition is a conjunction or disjunction.

For the experiments we have generated 100 ADF instances, with 20 instances of 10,15,20,25, and 30 statements. The instances have been generated via the random instance generator described in the previous paragraph with the default values of the generator for the parameters governing the ADF generation process: 7 for the width of the grid, 0.5 for the probability of symmetry in relation of attack or support, 0.2 for the probability of a variable being changed into a constant, and 0.5 for the probability of a given connective being a conjunction or disjunction when constructing the acceptance conditions.

For each ADF instance we also generated reasoning tasks for 3 arbitrarily chosen statements for each instance (the statements with identifiers 3, 5, and 7 in our instance set). This means that the total number of instances used in our experiments is 300. Based on first impressions we have also set a time-out for each computation (each run of `QADF` or `DIAMOND`) of 10 minutes (600 seconds). Computation times have been computed via the Unix `time` utility.

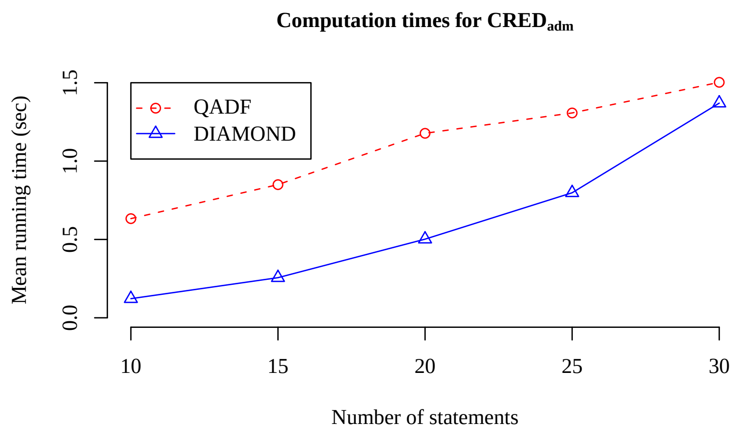
Results

In our experiments, there were no time-outs for either `DIAMOND` or `QADF` for CRED$_{adm}$, all computations taking below 2.5 seconds to complete. On the other hand, there were a significant number of time-outs for SKEPT$_{pref}$ for instances with above 20 statements. These are represented in Table 4.1 with respect to instance sets classified according to the number of statements the ADFs in them have (note that there are 60 instances per instance set).

Number of statements in ADFs	Time-outs `DIAMOND`	Time-outs `QADF`
10	0	0
15	0	0
20	30	25
25	54	55
30	45	59

Table 4.1: Number of time-outs for SKEPT$_{pref}$ (60 instances per instance set)

Figure 4.1 represents the mean running times of `QADF` and `DIAMOND` for CRED$_{adm}$ and SKEPT$_{pref}$ respectively, where the instances with time-outs (for SKEPT$_{pref}$) are disregarded (note in particular that for ADFs with 30 statements there is only one instance for which `QADF` does not time-out, while `DIAMOND` manages to solve 15 instances). The mean running time is plotted for the instance set of ADFs used for our experiments, partitioned according to the number of statements in the ADFs. The dotted line represents mean running time of QADF and the continuous line is for DIAMOND.



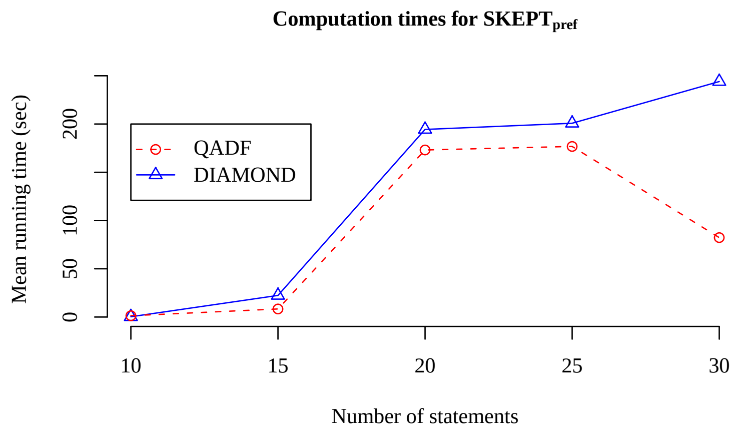
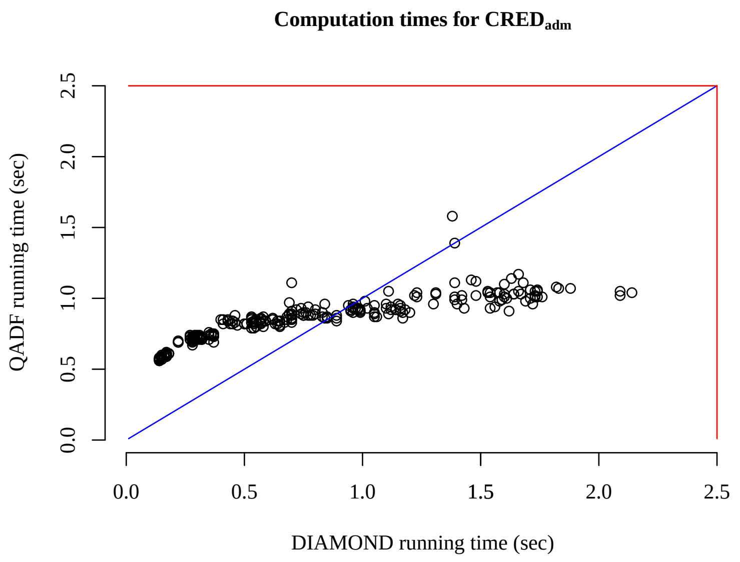


Figure 4.1: Mean running times of QADF and DIAMOND for CRED$_{adm}$ and SKEPT$_{pref}$

Finally, Figure 4.2 represents scatter plots for the computation times of QADF vs. DIAMOND for CRED$_{adm}$ and SKEPT$_{pref}$ respectively. In these graphs each dot corresponds to an instance in the experiments, the x-axis corresponding to the computation time of DIAMOND and the y-axis corresponding to the compuation time of QADF for that same instance. In this manner, the blue diagonal line in the middle of the graphs separates the instances for which DIAMOND has taken more time to compute (in the area below the line) from those instances for which QADF has taken more time to compute (area above the line).



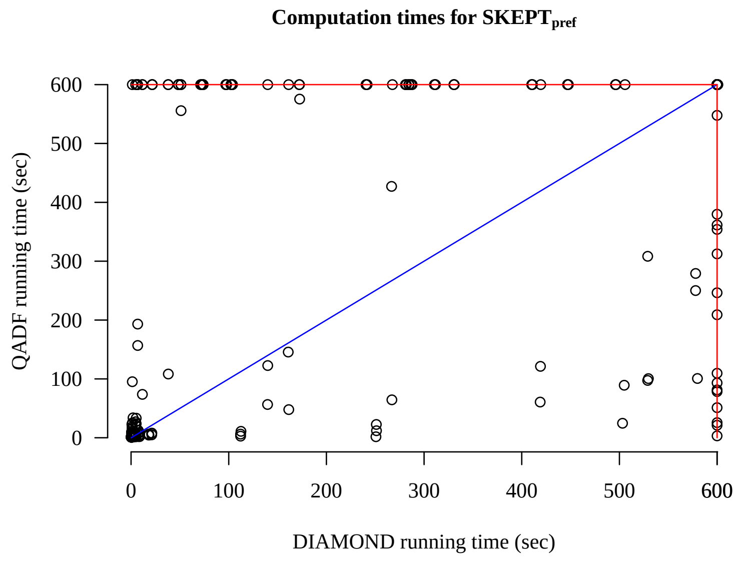


Figure 4.2: Scatter plots of running times of `QADF` and `DIAMOND` for CRED$_{adm}$ and SKEPT$_{pref}$

Discussion

The results presented in Section 4.4 suggest an acceptable performance of `QADF` for the reasoning task CRED$_{adm}$ on instances up to 30 statements, while there are clearly performance issues for SKEPT$_{pref}$ for ADF instances with already above 20 statements. This situation reflects the relative complexity of both tasks as presented in Section 2.4. The difference in complexity of both reasoning tasks could be considered to at least partially explain the corresponding difference in computation times, especially taking in account the similar situation for `DIAMOND`.

The results of the experiments in fact indicate no significant difference between `QADF` and `DIAMOND` (adapted by us to support skeptical and credulous reasoning), except for SKEPT$_{pref}$ for ADF instances with 30 statements, where `QADF` "timed-out" on all instances except one, while `DIAMOND` managed to solve 15 instances. This could suggest that the answer set programming approach (at least for preferred semantics) for solving reasoning problems on ADFs scales up better than the QBF-based approach (based on the encodings we present in this work), but given some limits in our experimental set-up and that this difference in performance is quite specific, more experiments would be necessary to reach a more definite conclusion about this issue.

In the first place, it should be observed that a limit of the current experimental set up is that because of the generator used, all the instances in the experiments are of a particular class of ADFs, the so called "bipolar ADFs" [Ellmauthaler, 2012]. A clearer picture of the performance of `QADF` (and `DIAMOND`) should be reached by dropping this restriction on the experimental set-up, although it is clearly not expected that the overall performance will improve (given that bipolar ADFs are a subset of ADFs with relatively "simple" structure).

A second issue is that, as hinted at in Section 4.2, there are quite a few different "flavors" of QSAT solvers available at the current date and there also various pre-processing techniques that have the potential of improving performance. Systematic testing of different combinations of QSAT solvers and pre-processing techniques together with `QADF` may lead to more insight on the scope and limits of an implementation of ADF-reasoning based on the encodings presented in this work. In particular, it may be interesting to try out QSAT solvers that do not require their input to be in PCNF. As a final note, it should also be observed that QSAT-technology is (also relative to answer set programming technology) somewhat in its infancy so it is reasonable to assume that further advances in this area will have a positive impact on QBF-based reasoners for ADFs.

CHAPTER 5

Conclusion and future work

In this work we have presented reductions of some of the main reasoning tasks defined for the relatively new and powerful formalism for abstract argumentation, ADFs, into important reasoning problems of the relatively well established formalism of quantified boolean logic. Specifically, we have given reductions for the valuation enumeration problem with respect to three valued models and two valued models, as well as admissible, complete, preferred, and grounded valuations into the problem of enumerating the models of a QBF. We have also provided reductions of the existence, non-trivial existence, and credulous as well as skeptical acceptance decision problems for ADFs with respect to the same semantics into the satisfiability problem of QBFs.

An important motivating factor for our work has been the strong link that exists between quantified boolean logic and the polynomial hierarchy, there being a prototypical QSAT problem for QBFs restricted to a certain prefix type for each level of the polynomial hierarchy. Given that the complexity of the decision problems we have considered in this work are complete for up to the third level of the polynomial hierarchy, this has allowed us to provide not only a uniform axiomatization of the reasoning tasks for ADFs in quantified boolean logic but also complexity sensitive encodings in the sense that the encodings have the prefix type corresponding to the level of the polynomial hierarchy for which the reasoning task they encode is complete. The fact that all of the encodings we present in this work express, in a relatively direct and "natural" manner, abstract necessary and sufficient conditions that capture the reasoning tasks we have investigated confirms the suitability of quantified boolean logic for the task we have set out to accomplish.

Moreover, we have been able to present the encodings in a modular fashion, also making repeated use of many of the modules. This suggests also that our encodings could be easily adapted either to provide encodings of new semantics (in so far as there is some relation between these and the semantics we consider in this work) for ADFs or restrict our encodings to certain special types of ADFs. This latter issue would be especially useful to investigate; in particular, it would be of considerable practical value (given that bipolar ADFs appear frequently in practical scenarios) to specialize our encodings to provide complexity-sensitive encodings of the reasoning tasks we have considered for bipolar ADFs.

An issue that may also be of some interest is to study the uses (for argumentation) and effect of extending ADFs by allowing many valued valuations (i.e. ADFs whose graph representation allows for acceptance conditions mapping parents of statements to more than two values). It is again plausible to assume that the general encoding strategy we present in this work could be easily adapted to (extensions of) semantics defined for these kinds of scenarios.

Turning to the more practical contributions of our work, by providing complexity-sensitive encodings for important reasoning tasks for ADFs we also provide a foundation for the implementation of reasoning systems for ADFs via QSAT solvers. Further on, we have carried out some first steps in assessing the convenience of this implementation strategy by having provided a prototype system based on the encodings presented in this work. We have also carried out preliminary experiments which show that reasoning on ADFs with up to 30 statments can be solved by our system within 2.5 seconds for credulous acceptance of statements with respect to admissible semantics. On the other hand, the results for skeptical reasoning with respect to preferred semantics, with high mean computation times and large amounts of time-outs for ADF instaces with more than 20 statments, are more discouraging.

The experiments on which we report on in this work suggest no clear advantage (nor disadvantage) of using the implementation approach based on quantified boolean logic (based on our encodings) to the answer set programming approach underlying the `DIAMOND` system. Nevertheless, further work would be necessary to assess more accurately the boundaries and scope of an implementation based on the encodings presented in this work. In particular, the experiments which we report on in this work are all based on instance sets containing only bipolar ADFs which are of a "lower complexity" and it would be useful to determine the relative performances of `DIAMOND` and `QADF` for ADFs not restricted to this type. Also, using a different combination of pre-processing technique and QSAT-solver may also lead to an improvement in performance, so it certainly would be of value to carry out experiments to determine if this is the case. In particular, it would be interesting to study if there is an improvement in the performance when making use of QSAT solvers which admit non-PCNF fromulas as input.

To conclude this work on an optimistic note, it should finally be observed that QSAT-solver technology is somewhat in its infancy and, hence, further advances in this area, which would also have a positive impact on implementations based on the encodings we present in this work, can be expected. In fact, the encodings we provide serve as interesting benchmarks that can contribute to the development of QSAT solvers.

Bibliography

[Amgoud et al., 2005] Amgoud, L., Belabbes, S., and Prade, H. (2005). Towards a Formal Framework for the Search of a Consensus Between Autonomous Agents. In Parsons, S., Maudet, N., Moraitis, P., and Rahwan, I., editors, *Proceedings of the Second International Workshop on Argumentation in Multi-Agent Systems (ArgMAS-2005)*, volume 4049 of *Lecture Notes in Computer Science*, pages 264–278. Springer.

[Amgoud and Cayrol, 2002] Amgoud, L. and Cayrol, C. (2002). Inferring from Inconsistency in Preference-Based Argumentation Frameworks. *Journal of Automated Reasoning*, 29(2):125–169.

[Amgoud and Devred, 2011] Amgoud, L. and Devred, C. (2011). Argumentation Frameworks as Constraint Satisfaction Problems. In Benferhat, S. and Grant, J., editors, *Proceedings of the Fifth International Conference on Scalable Uncertainty Management (SUM-2011)*, volume 6929 of *Lecture Notes in Computer Science*, pages 110–122. Springer.

[Amgoud and Prade, 2009] Amgoud, L. and Prade, H. (2009). Using Arguments for Making and Explaining Decisions. *Artificial Intelligence*, 173(3-4):413–436.

[Arieli, 2007] Arieli, O. (2007). Paraconsistent Reasoning and Preferential Entailments by Signed Quantified Boolean Formulae. *ACM Transations on Computuational Logic*, 8(3):1–29.

[Arieli and Caminada, 2012] Arieli, O. and Caminada, M. (2012). A General QBF-Based Formalization of Abstract Argumentation Theory. In Verheij, B., Szeider, S., and Woltran, S., editors, *Proceedings of the Fourth Conference on Computational Models of Argument (COMMA-2012)*, volume 245 of *Frontiers in Artificial Intelligence and Applications*, pages 105–116. IOS Press.

[Arieli and Caminada, 2013] Arieli, O. and Caminada, M. W. A. (2013). A QBF-Based Formalization of Abstract Argumentation Semantics. *Journal of Applied Logic*, 11(2):229–252.

[Arieli and Denecker, 2003] Arieli, O. and Denecker, M. (2003). Reducing Preferential Paraconsistent Reasoning to Classical Entailment. *Journal of Logic and Computation*, 13(4):557–580.

[Audemard and Sais, 2005] Audemard, G. and Sais, L. (2005). A Symbolic Search Based Approach for Quantified Boolean Formulas. In Bacchus, F. and Walsh, T., editors, *Proceedings of the Eighth International Conference on the Theory and Applications of Satisfiability Testing (SAT-2005)*, volume 3569 of *Lecture Notes in Computer Science*, pages 16–30. Springer.

[Ayari and Basin, 2002] Ayari, A. and Basin, D. A. (2002). QUBOS: Deciding Quantified Boolean Logic Using Propositional Satisfiability Solvers. In Aagaard, M. and O'Leary, J. W., editors, *Proceedings of the Fourth International Conference on Formal Methods in Computer-Aided Design (FMCAD-2002)*, volume 2517 of *Lecture Notes in Computer Science*, pages 187–201. Springer.

[Baroni et al., 2011a] Baroni, P., Caminada, M., and Giacomin, M. (2011a). An Introduction to Argumentation Semantics. *Knowledge Engineering Review*, 26(4):365–410.

[Baroni et al., 2011b] Baroni, P., Cerutti, F., Giacomin, M., and Guida, G. (2011b). AFRA: Argumentation Framework with Recursive Attacks. *International Journal of Approximate Reasoning*, 52(1):19–37.

[Baroni and Giacomin, 2007] Baroni, P. and Giacomin, M. (2007). On Principle-Based Evaluation of Extension-Based Argumentation Semantics. *Artificial Intelligence*, 171(10–15):675–700.

[Baroni and Giacomin, 2009] Baroni, P. and Giacomin, M. (2009). Semantics of Abstract Argument Systems. In Simari, G. and Rahwan, I., editors, *Argumentation in Artificial Intelligence*, pages 25–44. Springer.

[Bench-Capon et al., 2009] Bench-Capon, T., Prakken, H., and Sartor, G. (2009). Argumentation in Legal Reasoning. In Simari, G. and Rahwan, I., editors, *Argumentation in Artificial Intelligence*, pages 363–382. Springer.

[Bench-Capon, 2003] Bench-Capon, T. J. M. (2003). Persuasion in Practical Argument Using Value-Based Argumentation Frameworks. *Journal of Logic and Computation*, 13(3):429–448.

[Bench-Capon and Dunne, 2007] Bench-Capon, T. J. M. and Dunne, P. E. (2007). Argumentation in Artificial Intelligence. *Artificial Intelligence*, 171(10-15):619–641.

[Benedetti, 2005a] Benedetti, M. (2005a). Quantifier Trees for QBFs. In Bacchus, F. and Walsh, T., editors, *Proceedings of the Eighth International Conference on Theory and Applications of Satisfiability Testing (SAT-2005)*, volume 3569 of *Lecture Notes in Computer Science*, pages 378–385. Springer.

[Benedetti, 2005b] Benedetti, M. (2005b). sKizzo: A Suite to Evaluate and Certify QBFs. In Nieuwenhuis, R., editor, *Proceedings of the Twentieth International Conference on Automated Deduction (CADE-20)*, volume 3632 of *Lecture Notes in Computer Science*, pages 369–376. Springer.

[Benedetti and Mangassarian, 2008] Benedetti, M. and Mangassarian, H. (2008). QBF-Based Formal Verification: Experience and Perspectives. *Journal on Satisfiability, Boolean Modeling and Computation*, 5(1-4):133–191.

[Besnard and Doutre, 2004] Besnard, P. and Doutre, S. (2004). Checking the Acceptability of a Set of Arguments. In Delgrande, J. P. and Schaub, T., editors, *Proceedings of the Tenth International Workshop on Non-Monotonic Reasoning (NMR-2004)*, pages 59–64.

[Besnard and Hunter, 2005] Besnard, P. and Hunter, A. (2005). Practical First-Order Argumentation. In Veloso, M. M. and Kambhampati, S., editors, *Proceedings of the Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference (AAAI-2005)*, pages 590–595. AAAI Press / The MIT Press.

[Besnard and Hunter, 2008] Besnard, P. and Hunter, A. (2008). *Elements of Argumentation*. MIT Press.

[Besnard et al., 2005] Besnard, P., Schaub, T., Tompits, H., and Woltran, S. (2005). Representing Paraconsistent Reasoning via Quantified Propositional Logic. In Bertossi, L. E., Hunter, A., and Schaub, T., editors, *Inconsistency Tolerance*, volume 3300 of *Lecture Notes in Computer Science*, pages 84–118. Springer.

[Biere, 2004] Biere, A. (2004). Resolve and Expand. In Hoos, H. H. and Mitchell, D. G., editors, *Proceedings of the Seventh International Conference on Theory and Applications of Satisfiability Testing (SAT-2004)*, volume 3542 of *Lecture Notes in Computer Science*, pages 59–70. Springer.

[Biere, 2009] Biere, A. (2009). Bounded Model Checking. In Biere, A., Heule, M., van Maaren, H., and Walsh, T., editors, *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, pages 457–481. IOS Press.

[Biere et al., 2011] Biere, A., Lonsing, F., and Seidl, M. (2011). Blocked Clause Elimination for QBF. In Bjørner, N. and Sofronie-Stokkermans, V., editors, *Proceedings of the Twenty-Third International Conference on Automated Deduction (CADE-11)*, volume 6803 of *Lecture Notes in Computer Science*, pages 101–115. Springer.

[Bistarelli and Santini, 2011] Bistarelli, S. and Santini, F. (2011). ConArg: A Constraint-Based Computational Framework for Argumentation Systems. In Khoshgoftaar, T. M. and Zhu, X. H., editors, *Proceedings of the Twenty-Third IEEE International Conference on Tools with Artificial Intelligence (ICTAI-2011)*, pages 605–612. IEEE Computer Society Press.

[Blackburn et al., 2001] Blackburn, P., de Rijke, M., and Venema, Y. (2001). *Modal Logic*, volume 53 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press.

[Bondarenko et al., 1997] Bondarenko, A., Dung, P. M., Kowalski, R. A., and Toni, F. (1997). An Abstract, Argumentation-Theoretic Approach to Default Reasoning. *Artificial Intelligence*, 93(1):63–101.

[Bondarenko et al., 1993] Bondarenko, A., Toni, F., and Kowalski, R. A. (1993). An Assumption-Based Framework for Non-Monotonic Reasoning. In Pereira, L. M. and Nerode, A., editors, *Proceedings of the Second International Workshop on Logic Programming and Non-Monotonic Reasoning (LPNMR-1993)*, pages 171–189. The MIT Press.

[Brewka et al., 2011] Brewka, G., Eiter, T., and Truszczyński, M. (2011). Answer Set Programming at a Glance. *Communications of the ACM*, 54(12):92–103.

[Brewka et al., 2013] Brewka, G., Ellmauthaler, S., Strass, H., Wallner, J. P., and Woltran, S. (2013). Abstract Dialectical Frameworks Revisited. In Rossi, F., editor, *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence (IJCAI-2013)*, pages 803–809. AAAI Press / IJCAI.

[Brewka and Woltran, 2010] Brewka, G. and Woltran, S. (2010). Abstract Dialectical Frameworks. In Lin, F., Sattler, U., and Truszczynski, M., editors, *Proceedings of the Twelfth International Conference on the Principles of Knowledge Representation and Reasoning (KR-2010)*. AAAI Press.

[Bryant et al., 2003] Bryant, R. E., Lahiri, S. K., and Seshia, S. A. (2003). Convergence Testing in Term-Level Bounded Model Checking. In Geist, D. and Tronci, E., editors, *Proceedings of the Twelfth Advanced Research Working Conference on Correct Hardware Design and Verification Methods (CHARME-2003)*, volume 2860 of *Lecture Notes in Computer Science*, pages 348–362. Springer.

[Bubeck, 2010] Bubeck, U. (2010). *Model-Based Transformations for Quantified Boolean Formulas*. PhD thesis, Universität Paderborn.

[Bubeck and Büning, 2007] Bubeck, U. and Büning, H. K. (2007). Bounded Universal Expansion for Preprocessing QBF. In Marques-Silva, J. and Sakallah, K. A., editors, *Procedings of the Tenth International Conference on the Theory and Applications of Satisfiability Testing (SAT-2007)*, volume 4501 of *Lecture Notes in Computer Science*, pages 244–257. Springer.

[Büning and Bubeck, 2009] Büning, H. K. and Bubeck, U. (2009). Theory of Quantified Boolean Formulas. In Biere, A., Heule, M., van Maaren, H., and Walsh, T., editors, *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, pages 735–760. IOS Press.

[Büning et al., 1995] Büning, H. K., Karpinski, M., and Flögel, A. (1995). Resolution for Quantified Boolean Formulas. *Information and Computation*, 117(1):12–18.

[Büning and Letterman, 1999] Büning, H. K. and Letterman, T. (1999). *Propositional Logic: Deduction and Algorithms*. Cambridge University Press.

[Cadoli et al., 1998] Cadoli, M., Giovanardi, A., and Schaerf, M. (1998). An Algorithm to Evaluate Quantified Boolean Formulae. In Mostow, J. and Rich, C., editors, *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, pages 262–267. AAAI Press/MIT Press.

[Caminada and Amgoud, 2007] Caminada, M. and Amgoud, L. (2007). On the Evaluation of Argumentation Formalisms. *Artificial Intelligence*, 171(5-6):286–310.

[Caminada and Gabbay, 2009] Caminada, M. W. A. and Gabbay, D. M. (2009). A Logical Account of Formal Argumentation. *Studia Logica*, 93(2-3):109–145.

[Cartwright and Atkinson, 2009] Cartwright, D. and Atkinson, K. (2009). Using Computational Argumentation to Support E-participation. *IEEE Intelligent Systems*, 24(5):42–52.

[Cayrol and Lagasquie-Schiex, 2005] Cayrol, C. and Lagasquie-Schiex, M. (2005). On the Acceptability of Arguments in Bipolar Argumentation Frameworks. In Godo, L., editor, *Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, volume 3571 of *Lecture Notes in Computer Science*, pages 378–389. Springer.

[Cerutti et al., 2013] Cerutti, F., Dunne, P. E., Giacomin, M., and Vallati, M. (2013). Computing Preferred Extensions in Abstract Argumentation: A SAT-Based Approach. In Black, E., Modgil, S., and Oren, N., editors, *Proceedings of the Second International Workshop of Theory and Applications of Formal Argumentation (TAFA-2013)*, volume 8306 of *Lecture Notes in Computer Science*, pages 176–193. Springer.

[Charwat et al., 2013] Charwat, G., Dvořák, W., Gaggl, S. A., Wallner, J. P., and Woltran, S. (2013). Implementing Abstract Argumentation - A Survey. Technical Report DBAI-TR-2013-82, Technische Universität Wien.

[Chesñevar et al., 2000] Chesñevar, C. I., Maguitman, A. G., and Loui, R. P. (2000). Logical Models of Argument. *ACM Computing Surveys*, 32(4):337–383.

[Cook, 1971] Cook, S. (1971). The Complexity of Theorem-Proving Procedures. In *Conference Record of Third Annual ACM Symposium on Theory of Computing (STOC-71)*, pages 151–158.

[Coste-Marquis et al., 2005] Coste-Marquis, S., Devred, C., and Marquis, P. (2005). Symmetric Argumentation Frameworks. In Godo, L., editor, *Proceedings of the Eighth European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU-2005)*, volume 3571 of *Lecture Notes in Computer Science*, pages 317–328. Springer.

[Coste-Marquis et al., 2006] Coste-Marquis, S., Devred, C., and Marquis, P. (2006). Constrained Argumentation Frameworks. In Doherty, P., Mylopoulos, J., and Welty, C. A., editors, *Proceedings of the Tenth International Conference on Principles of Knowledge Representation and Reasoning (KR-2006)*, pages 112–122. AAAI Press.

[Davis et al., 1962] Davis, M., Logemann, G., and Loveland, D. (1962). A Machine Program for Theorem Proving. *Communications of the ACM*, 5(7):394–397.

[Davis and Putnam, 1960] Davis, M. and Putnam, H. (1960). A Computing Procedure for Quantification Theory. *Journal of the ACM*, 7(3):201–215.

[Dimopoulos and Torres, 1996] Dimopoulos, Y. and Torres, A. (1996). Graph Theoretical Structures in Logic Programs and Default Theories. *Theoretical Computer Science*, 170(1-2):209–244.

[Donini et al., 2002] Donini, F. M., Liberatore, P., Massacci, F., and Schaerf, M. (2002). Solving QBF by SMV. In Fensel, D., Giunchiglia, F., McGuinness, D. L., and Williams, M.-A., editors, *Proceedings of the Eighth International Conference on Principles of Knowledge Representation and Reasoning (KR-02)*, pages 578–592. Morgan Kaufmann.

[Doutre and Mengin, 2001] Doutre, S. and Mengin, J. (2001). Preferred Extensions of Argumentation Frameworks: Query Answering and Computation. In Goré, R., Leitsch, A., and Nipkow, T., editors, *Proceedings of the First International Joint Conference on Automated Reasoning (IJCAR-2001)*, volume 2083 of *Lecture Notes in Computer Science*, pages 272–288. Springer.

[Dung, 1995] Dung, P. M. (1995). On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and N-person Games. *Artificial Intelligence*, 77(2):321–357.

[Dunne and Bench-Capon, 2002] Dunne, P. E. and Bench-Capon, T. J. M. (2002). Coherence in Finite Argument Systems. *Artificial Intelligence*, 141(1/2):187–203.

[Dvorák et al., 2011] Dvorák, W., Morak, M., Nopp, C., and Woltran, S. (2011). dynPARTIX - A Dynamic Programming Reasoner for Abstract Argumentation. In Tompits, H., Abreu, S., Oetsch, J., Pührer, J., Seipel, D., Umeda, M., and Wolf, A., editors, *Proceedings of the Nineteenth International Conference on the Applications of Declarative Programming and Knowledge Management (INAP-2011) and Twenty-Fifth Workshop on Logic Programming (WLP-2011)*, volume 7773 of *Lecture Notes in Computer Science*, pages 259–268. Springer.

[Dvořák et al., 2012] Dvořák, W., Järvisalo, M., Wallner, J. P., and Woltran, S. (2012). Complexity-Sensitive Decision Procedures for Abstract Argumentation. In Brewka, G., Eiter, T., and McIlraith, S. A., editors, *Proceedings of the Thirteenth International Conference on Principles of Knowledge Representation and Reasoning (KR-2012)*, pages 54–64. AAAI Press.

[Dvořák et al., 2012a] Dvořák, W., Pichler, R., and Woltran, S. (2012a). Towards Fixed-Parameter Tractable Algorithms for Abstract Argumentation. *Artificial Intelligence*, 186:1–37.

[Dvořák et al., 2012b] Dvořák, W., Szeider, S., and Woltran, S. (2012b). Abstract Argumentation via Monadic Second Order Logic. In Hüllermeier, E., Link, S., Fober, T., and Seeger, B., editors, *Proceedings of the Sixth International Conference on Scalable Uncertainty Management (SUM-2012)*, volume 7520 of *Lecture Notes in Computer Science*, pages 85–98. Springer.

[Dvořák and Woltran, 2011] Dvořák, W. and Woltran, S. (2011). On the Intertranslatability of Argumentation Semantics. *Journal of Artificial Intelligence Research*, 41:445–475.

[Eén and Biere, 2005] Eén, N. and Biere, A. (2005). Effective Preprocessing in SAT Through Variable and Clause Elimination. In Bacchus, F. and Walsh, T., editors, *Proceedings of the Eighth International Conference on Theory and Applications of Satisfiability Testing (SAT-2005)*, volume 3569 of *Lecture Notes in Computer Science*, pages 61–75. Springer.

[Egly et al., 2000] Egly, U., Eiter, T., Tompits, H., and Woltran, S. (2000). Solving Advanced Reasoning Tasks Using Quantified Boolean Formulas. In Kautz, H. A. and Porter, B. W., editors, *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on on Innovative Applications of Artificial Intelligence (AAAI/IAAI-2000)*, pages 417–422. AAAI Press / The MIT Press.

[Egly et al., 2010] Egly, U., Gaggl, S. A., and Woltran, S. (2010). Answer-Set Programming Encodings for Argumentation Frameworks. *Argument & Computation*, 1(2):147–177.

[Egly et al., 2006] Egly, U., Seidl, M., and Woltran, S. (2006). A Solver for QBFs in Non-prenex Form. In Brewka, G., Coradeschi, S., Perini, A., and Traverso, P., editors, *Proceedings of the Seventeenth European Conference on Artificial Intelligence Including Prestigious Applications of Intelligent Systems (ECAI/PAIS-2006)*, volume 141 of *Frontiers in Artificial Intelligence and Applications*, pages 477–481. IOS Press.

[Egly et al., 2009] Egly, U., Seidl, M., and Woltran, S. (2009). A Solver for QBFs in Negation Normal Form. *Constraints*, 14(1):38–79.

[Egly and Woltran, 2006] Egly, U. and Woltran, S. (2006). Reasoning in Argumentation Frameworks Using Quantified Boolean Formulas. In Dunne, P. E. and Bench-Capon, T. J. M., editors, *Proceedings of the First Conference on Computational Models of Argument (COMMA-2006)*, volume 144 of *Frontiers in Artificial Intelligence and Applications*, pages 133–144. IOS Press.

[Eiter and Gottlob, 1995] Eiter, T. and Gottlob, G. (1995). On the Computational Cost of Disjunctive Logic Programming: Propositional Case. *Annals of Mathematics and Artificial Intelligence*, 15(3-4):289–323.

[Ellmauthaler, 2012] Ellmauthaler, S. (2012). Abstract Dialectical Frameworks: Properties, Complexity, and Implementation. Master's thesis, Technische Universität Wien, Institut für Informationssysteme.

[Ellmauthaler and Strass, 2013] Ellmauthaler, S. and Strass, H. (2013). The DIAMOND System for Argumentation: Preliminary Report. In Fink, M. and Lierler, Y., editors, *Proceedings of the Sixth International Workshop on Answer Set Programming and Other Computing Paradigms (ASPOCP-2013)*, pages 97–108.

[Ellmauthaler and Wallner, 2012] Ellmauthaler, S. and Wallner, J. P. (2012). Evaluating Abstract Dialectical Frameworks with ASP. In Verheij, B., Szeider, S., and Woltran, S., editors, *Proceedings of the Fourth International Conference on Computational Models of Argument (COMMA-2012)*, volume 245 of *Frontiers in Artificial Intelligence and Applications*, pages 505–506. IOS Press.

[Feldmann et al., 2000] Feldmann, R., Monien, B., and Schamberger, S. (2000). A Distributed Algorithm to Evaluate Quantified Boolean Formulae. In Kautz, H. A. and Porter, B. W., editors, *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on on Innovative Applications of Artificial Intelligence (AAAI/IAAI-2000)*, pages 285–290. AAAI Press / The MIT Press.

[Ferraris and Giunchiglia, 2000] Ferraris, P. and Giunchiglia, E. (2000). Planning as Satisfiability in Nondeterministic Domains. In Kautz, H. A. and Porter, B. W., editors, *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence (AAAI/IAAI-2000)*, pages 748–753. AAAI Press / The MIT Press.

[Gabbay, 2011] Gabbay, D. M. (2011). Dung's Argumentation is Essentially Equivalent to Classical Propositional Logic with the Peirce-Quine Dagger. *Logica Universalis*, 5(2):255–318.

[Gabbay, 2012a] Gabbay, D. M. (2012a). An Equational Approach to Argumentation Networks. *Argument & Computation*, 3(2-3):87–142.

[Gabbay, 2012b] Gabbay, D. M. (2012b). The Equational Approach to CF2 Semantics. In Verheij, B., Szeider, S., and Woltran, S., editors, *Proceedings of the Fourth Conference on Computational Models of Argument (COMMA-2012)*, volume 245 of *Frontiers in Artificial Intelligence and Applications*, pages 141–152. IOS Press.

[Garey and Johnson, 1979] Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability*. W. H. Freeman.

[Gebser et al., 2011] Gebser, M., Kaufmann, B., Kaminski, R., Ostrowski, M., Schaub, T., and Schneider, M. T. (2011). Potassco: The Potsdam Answer Set Solving Collection. *AI Communications*, 24(2):107–124.

[Gelder et al., 2012] Gelder, A. V., Wood, S. B., and Lonsing, F. (2012). Extended Failed-Literal Preprocessing for Quantified Boolean Formulas. In Cimatti, A. and Sebastiani, R., editors, *Proceedings of the Fifteenth International Conference on Theory and Applications of Satisfiability Testing (SAT-2012)*, volume 7317 of *Lecture Notes in Computer Science*, pages 86–99. Springer.

[Gelfond and Lifschitz, 1988] Gelfond, M. and Lifschitz, V. (1988). The Stable Model Semantics for Logic Programming. In Kowalski, R. A. and Bowen, K. A., editors, *Proceedings of the Fifth International Conference on Logic Programming (ICLP-1988)*, pages 1070–1080. MIT Press.

[Giunchiglia et al., 2009] Giunchiglia, E., Marin, P., and Narizzano, M. (2009). Reasoning with Quantified Boolean Formulas. In Biere, A., Heule, M., van Maaren, H., and Walsh, T., editors, *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, pages 761–780. IOS Press.

[Giunchiglia et al., 2010a] Giunchiglia, E., Marin, P., and Narizzano, M. (2010a). QuBE7.0. *Journal on Satisfiability, Boolean Modeling and Computation*, 7(2-3):83–88.

[Giunchiglia et al., 2010b] Giunchiglia, E., Marin, P., and Narizzano, M. (2010b). sQueezeBF: An Effective Preprocessor for QBFs Based on Equivalence Reasoning. In Strichman, O. and Szeider, S., editors, *Proceedings of the Thirteenth International Conference on Theory and Applications of Satisfiability Testing (SAT-2010)*, volume 6175 of *Lecture Notes in Computer Science*, pages 85–98. Springer.

[Giunchiglia et al., 2001a] Giunchiglia, E., Narizzano, M., and Tacchella, A. (2001a). An Analysis of Backjumping and Trivial Truth in Quantified Boolean Formulas Satisfiability. In Esposito, F., editor, *Proceedings of the Seventh Congress of the Italian Association for Artificial Intelligence (AI*IA-2001)*, volume 2175 of *Lecture Notes in Computer Science*, pages 111–122. Springer.

[Giunchiglia et al., 2001b] Giunchiglia, E., Narizzano, M., and Tacchella, A. (2001b). QUBE: A System for Deciding Quantified Boolean Formulas Satisfiability. In Goré, R., Leitsch, A., and Nipkow, T., editors, *Proceedings of the First International Joint Conference on Automated Reasoning (IJCAR-2001)*, volume 2083 of *Lecture Notes in Computer Science*, pages 364–369. Springer.

[Giunchiglia et al., 2002] Giunchiglia, E., Narizzano, M., and Tacchella, A. (2002). Learning for Quantified Boolean Logic Satisfiability. In Dechter, R. and Sutton, R. S., editors, *Proceedings of the Eighteenth National Conference on Artificial Intelligence and Fourteenth Conference on Innovative Applications of Artificial Intelligence (AAAI-2002)*, pages 649–654. AAAI Press / The MIT Press.

[Giunchiglia et al., 2007] Giunchiglia, E., Narizzano, M., and Tacchella, A. (2007). Quantifier Structure in Search-Based Procedures for QBFs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 26(3):497–507.

[Goultiaeva and Bacchus, 2010] Goultiaeva, A. and Bacchus, F. (2010). Exploiting QBF Duality on a Circuit Representation. In Fox, M. and Poole, D., editors, *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI-2010)*. AAAI Press.

[Goultiaeva and Bacchus, 2013] Goultiaeva, A. and Bacchus, F. (2013). Recovering and Utilizing Partial Duality in QBF. In Järvisalo, M. and Gelder, A. V., editors, *Proceedings of the Sixteenth International Conference on Theory and Applications of Satisfiability Testing (SAT-2013)*, volume 7962 of *Lecture Notes in Computer Science*, pages 83–99. Springer.

[Goultiaeva et al., 2009] Goultiaeva, A., Iverson, V., and Bacchus, F. (2009). Beyond CNF: A Circuit-Based QBF Solver. In Kullmann, O., editor, *Proceedings of the Twelfth International Conference on Theory and Applications of Satisfiability Testing (SAT-2009)*, volume 5584 of *Lecture Notes in Computer Science*, pages 412–426. Springer.

[Hölldobler, 2011] Hölldobler, S. (2011). *Logik und Logikprogrammierung: Band 1: Grundlagen*. Kolleg Synchron. Synchron.

[Huth and Ryan, 2004] Huth, M. and Ryan, M. (2004). *Logic in Computer Science: Modelling and Reasoning About Systems*. Cambridge University Press.

[Janota et al., 2012] Janota, M., Klieber, W., Marques-Silva, J., and Clarke, E. M. (2012). Solving QBF with Counterexample Guided Refinement. In Cimatti, A. and Sebastiani, R., editors, *Proceedings of the Fifteenth International Conference on the Theory and Applications of Satisfiability Testing (SAT-2012)*, volume 7317 of *Lecture Notes in Computer Science*, pages 114–128. Springer.

[Järvisalo and Gelder, 2013] Järvisalo, M. and Gelder, A. V., editors (2013). *Proceedings of the Sixteenth International Conference on Theory and Applications of Satisfiability Testing (SAT 2013)*, volume 7962 of *Lecture Notes in Computer Science*. Springer.

[Johnson and Blair, 1994] Johnson, R. and Blair, J. (1994). *Logical Self-Defense*. McGraw Hill-Ryerson.

[Jussila et al., 2007] Jussila, T., Biere, A., Sinz, C., Kröning, D., and Wintersteiger, C. M. (2007). A First Step Towards a Unified Proof Checker for QBF. In Marques-Silva, J. and Sakallah, K. A., editors, *Proceedings of the Tenth International Conference on Theory and Applications of Satisfiability Testing (SAT-2007)*, volume 4501 of *Lecture Notes in Computer Science*, pages 201–214. Springer.

[Jussila et al., 2006] Jussila, T., Sinz, C., and Biere, A. (2006). Extended Resolution Proofs for Symbolic SAT Solving with Quantification. In Biere, A. and Gomes, C. P., editors, *Proceedings of the Ninth International Conference on Theory and Applications of Satisfiability Testing (SAT-2006)*, volume 4121 of *Lecture Notes in Computer Science*, pages 54–60. Springer.

[Kakas and Moraitis, 2006] Kakas, A. C. and Moraitis, P. (2006). Adaptive Agent Negotiation via Argumentation. In Nakashima, H., Wellman, M. P., Weiss, G., and Stone, P., editors, *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2006)*, pages 384–391. ACM.

[Kautz and Selman, 1992] Kautz, H. A. and Selman, B. (1992). Planning as Satisfiability. In Neumann, B., editor, *Proceedings of the Tenth European Conference on Artificial Intelligence (ECAI-1992)*, pages 359–363. John Wiley and Sons.

[Kleene, 1952] Kleene, S. (1952). *Introduction to Metamathematics*. Van Nostrand.

[Klement, 2013] Klement, K. C. (2013). Propositional Logic. `http://www.iep.utm.edu/prop-log/`. Accessed: 2013-10-04.

[Klieber et al., 2010] Klieber, W., Sapra, S., Gao, S., and Clarke, E. M. (2010). A Non-prenex, Non-clausal QBF Solver with Game-State Learning. In Strichman, O. and Szeider, S., editors, *Proceedings of the Thirteenth International Conference on Theory and Applications of Satisfiability Testing (SAT-2010)*, volume 6175 of *Lecture Notes in Computer Science*, pages 128–142. Springer.

[Kroening, 2009] Kroening, D. (2009). Software Verification. In Biere, A., Heule, M., van Maaren, H., and Walsh, T., editors, *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, pages 505–532. IOS Press.

[Kroening and Strichman, 2008] Kroening, D. and Strichman, O. (2008). *Decision Procedures: An Algorithmic Point of View*. Springer.

[Leone et al., 2006] Leone, N., Pfeifer, G., Faber, W., Eiter, T., Gottlob, G., Perri, S., and Scarcello, F. (2006). The DLV System for Knowledge Representation and Reasoning. *ACM Transactions on Computational Logic*, 7(3):499–562.

[Letz, 2002] Letz, R. (2002). Lemma and Model Caching in Decision Procedures for Quantified Boolean Formulas. In Egly, U. and Fermüller, C. G., editors, *Proceedings of the International Conference on Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX-2002)*, volume 2381 of *Lecture Notes in Computer Science*, pages 160–175. Springer.

[Lewis et al., 2009] Lewis, M. D. T., Marin, P., Schubert, T., Narizzano, M., Becker, B., and Giunchiglia, E. (2009). PaQuBE: Distributed QBF Solving with Advanced Knowledge Sharing. In Kullmann, O., editor, *Proceedings of the Twelfth Conference on Theory and Applications of Satisfiability Testing (SAT-2009)*, volume 5584 of *Lecture Notes in Computer Science*, pages 509–523. Springer.

[Lewis et al., 2011] Lewis, M. D. T., Schubert, T., Becker, B., Marin, P., Narizzano, M., and Giunchiglia, E. (2011). Parallel QBF Solving with Advanced Knowledge Sharing. *Fundamenta Informaticae*, 107(2-3):139–166.

[Ling et al., 2005] Ling, A. C., Singh, D. P., and Brown, S. D. (2005). FPGA Logic Synthesis Using Quantified Boolean Satisfiability. In Bacchus, F. and Walsh, T., editors, *Proceedings of the Eighth International Conference on Theory and Applications of Satisfiability Testing (SAT-2005)*, volume 3569 of *Lecture Notes in Computer Science*, pages 444–450. Springer.

[Lonsing, 2012] Lonsing, F. (2012). *Dependency Schemes and Search-Based QBF Solving: Theory and Practice*. PhD thesis, Johannes Kepler Universität, Linz.

[Lonsing and Biere, 2008] Lonsing, F. and Biere, A. (2008). Nenofex: Expanding NNF for QBF Solving. In Büning, H. K. and Zhao, X., editors, *Proceedings of the Eleventh International Conference on Theory and Applications of Satisfiability Testing (SAT-2008)*, volume 4996 of *Lecture Notes in Computer Science*, pages 196–210. Springer.

[Lonsing and Biere, 2010a] Lonsing, F. and Biere, A. (2010a). DepQBF: A Dependency-Aware QBF Solver. *Journal on Satisfiability, Boolean Modelling and Computation*, 7(2-3):71–76.

[Lonsing and Biere, 2010b] Lonsing, F. and Biere, A. (2010b). Integrating Dependency Schemes in Search-Based QBF Solvers. In Strichman, O. and Szeider, S., editors, *Proceedings of the Thirteenth International Conference on Theory and Applications of Satisfiability Testing (SAT 2010)*, volume 6175 of *Lecture Notes in Computer Science*, pages 158–171. Springer.

[Lonsing and Biere, 2011] Lonsing, F. and Biere, A. (2011). Failed Literal Detection for QBF. In Sakallah, K. A. and Simon, L., editors, *Proceedings of the Fourteenth International Conference on Theory and Applications of Satisfiability Testing (SAT-2011)*, volume 6695 of *Lecture Notes in Computer Science*, pages 259–272. Springer.

[Lonsing and Seidl, 2013] Lonsing, F. and Seidl, M., editors (2013). *Informal Workshop Report on the International Workshop on Quantified Boolean Formulas 2013.*

[McBurney et al., 2012] McBurney, P., Parsons, S., and Rahwan, I., editors (2012). *Proceedings of the Eighth International Workshop on Argumentation in Multi-Agent Systems (ArgMAS-2011)*, volume 7543 of *Lecture Notes in Computer Science*. Springer.

[McMillan, 1993] McMillan, K. L. (1993). *Symbolic model checking*. Kluwer.

[Mneimneh and Sakallah, 2003] Mneimneh, M. N. and Sakallah, K. A. (2003). Computing Vertex Eccentricity in Exponentially Large Graphs: QBF Formulation and Solution. In Giunchiglia, E. and Tacchella, A., editors, *Proceedings of the Sixth International Conference on Theory and Applications of Satisfiability Testing (SAT-2003)*, volume 2919 of *Lecture Notes in Computer Science*, pages 411–425. Springer.

[Modgil and Caminada, 2009] Modgil, S. and Caminada, M. (2009). Proof Theories and Algorithms for Abstract Argumentation Frameworks. In Rahwan, I. and Simari, G. R., editors, *Argumentation in Artificial Intelligence*, pages 105–132. Springer.

[Nieves et al., 2008] Nieves, J. C., Osorio, M., and Cortés, U. (2008). Preferred Extensions as Stable Models. *Theory and Practice of Logic Programming*, 8(4):527–543.

[Nofal et al., 2012] Nofal, S., Dunne, P. E., and Atkinson, K. (2012). On Preferred Extension Enumeration in Abstract Argumentation. In Verheij, B., Szeider, S., and Woltran, S., editors, *Proceedings of the Fourth Conference on Computational Models of Argument (COMMA-2012)*, volume 245 of *Frontiers in Artificial Intelligence and Applications*, pages 205–216. IOS Press.

[Odersky et al., 2011] Odersky, M., Spoon, L., and Venners, B. (2011). *Programming in Scala: A Comprehensive Step-by-Step Guide, 2nd Edition*. Artima Incorporation.

[Olivo and Emerson, 2011] Olivo, O. and Emerson, E. A. (2011). A More Efficient BDD-Based QBF Solver. In Lee, J. H.-M., editor, *Proceedings of the Seventeenth International Conference on Principles and Practice of Constraint Programming (CP-2011)*, volume 6876 of *Lecture Notes in Computer Science*, pages 675–690. Springer.

[Pan and Vardi, 2004] Pan, G. and Vardi, M. Y. (2004). Symbolic Decision Procedures for QBF. In Wallace, M., editor, *Proceedings of the Tenth International Conference on Principles and Practice of Constraint Programming (CP-2004)*, volume 3258 of *Lecture Notes in Computer Science*, pages 453–467. Springer.

[Peschiera et al., 2010] Peschiera, C., Pulina, L., Tacchella, A., Bubeck, U., Kullmann, O., and Lynce, I. (2010). The Seventh QBF Solvers Evaluation (QBFEVAL'10). In Strichman, O. and Szeider, S., editors, *Proceedings of the Thirteenth International Conference on the Theory and Applications of Satisfiability Testing (SAT-2010)*, volume 6175 of *Lecture Notes in Computer Science*, pages 237–250. Springer.

[Pigorsch and Scholl, 2009] Pigorsch, F. and Scholl, C. (2009). Exploiting Structure in an AIG Based QBF Solver. In *Proceedings of the Conference on Design, Automation and Test in Europe (DATE-2009)*, pages 1596–1601. IEEE.

[Podlaszewski et al., 2011] Podlaszewski, M., Caminada, M., and Pigozzi, G. (2011). An Implementation of Basic Argumentation Components. In Sonenberg, L., Stone, P., Tumer, K., and Yolum, P., editors, *Proceedings of the Tenth International Conference on Autonomous Agents and Multiagent Systems (AAMAS-2011)*, pages 1307–1308. IFAAMAS.

[Poe, 2006] Poe, E. A. (2006). *The Murders in the Rue Morgue*. Modern Library.

[Pollock, 1987] Pollock, J. L. (1987). Defeasible Reasoning. *Cognitive Science*, 11(4):481–518.

[Prakken and Sartor, 1997] Prakken, H. and Sartor, G. (1997). Argument-Based Extended Logic Programming with Defeasible Priorities. *Journal of Applied Non-Classical Logics*, 7(1):25–75.

[Prakken and Vreeswijk, 2002] Prakken, H. and Vreeswijk, G. (2002). Logics for Defeasible Argumentation. In Gabbay, D. and Guenthner, F., editors, *Handbook of Philosophical Logic, Second Edition, Vol. 4*, pages 219–318. Dordrecht etc.

[Pulina and Tacchella, 2007] Pulina, L. and Tacchella, A. (2007). A Multi-Engine Solver for Quantified Boolean Formulas. In Bessiere, C., editor, *Proceedings of the Thirteenth International Conference on Principles and Practice of Constraint Programming (CP-2007)*, volume 4741 of *Lecture Notes in Computer Science*, pages 574–589. Springer.

[Rahwan and Simari, 2009] Rahwan, I. and Simari, G. R. (2009). *Argumentation in Artificial Intelligence*. Springer.

[Reimer et al., 2011] Reimer, S., Pigorsch, F., Scholl, C., and Becker, B. (2011). Integration of Orthogonal QBF Solving Techniques. In *Proceedings of Conference on Design, Automation and Test in Europe (DATE-2011)*, pages 149–154. IEEE.

[Reiter, 1980] Reiter, R. (1980). A Logic for Default Reasoning. *Artificial Intelligence*, 13(1-2):81–132.

[Rintanen, 1999a] Rintanen, J. (1999a). Constructing Conditional Plans by a Theorem-Prover. *Journal of Artificial Intellegence Research*, 10:323–352.

[Rintanen, 1999b] Rintanen, J. (1999b). Improvements to the Evaluation of Quantified Boolean Formulae. In Dean, T., editor, *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99)*, pages 1192–1197. Morgan Kaufmann.

[Rintanen, 2009] Rintanen, J. (2009). Planning and SAT. In Biere, A., Heule, M., van Maaren, H., and Walsh, T., editors, *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, pages 483–504. IOS Press.

[Robinson, 1965] Robinson, J. A. (1965). A Machine-Oriented Logic Based on the Resolution Principle. *Journal of the ACM*, 12(1):23–41.

[Samer and Szeider, 2009] Samer, M. and Szeider, S. (2009). Backdoor Sets of Quantified Boolean Formulas. *Journal of Automated Reasoning*, 42(1):77–97.

[Simons et al., 2002] Simons, P., Niemelä, I., and Soininen, T. (2002). Extending and Implementing the Stable Model Semantics. *Artificial Intelligence*, 138(1-2):181–234.

[South et al., 2008] South, M., Vreeswijk, G., and Fox, J. (2008). Dungine: A Java Dung Reasoner. In Besnard, P., Doutre, S., and Hunter, A., editors, *Proceedings of the Second Conference on Computational Models of Argument (COMMA-2008)*, volume 172 of *Frontiers in Artificial Intelligence and Applications*, pages 360–368. IOS Press.

[Stockmeyer, 1976] Stockmeyer, L. J. (1976). The Polynomial-Time Hierarchy. *Theoretical Computer Science*, 31(1):1–22.

[Stockmeyer and Meyer, 1973] Stockmeyer, L. J. and Meyer, A. R. (1973). Word Problems Requiring Exponential Time (Preliminary Report). In Aho, A. V., Borodin, A., Constable, R. L., Floyd, R. W., Harrison, M. A., Karp, R. M., and Strong, H. R., editors, *Proceedings of the Fifth Annual ACM Symposium on Theory of Computing (STOC-1973)*, pages 1–9. ACM.

[Strass, 2013] Strass, H. (2013). Approximating Operators and Semantics for Abstract Dialectical Frameworks. *Artificial Intelligence*, 205:39–70.

[Strass and Wallner, 2013] Strass, H. and Wallner, J. P. (2013). Analyzing the Computational Complexity of Abstract Dialectical Frameworks via Approximation Fixpoint Theory. Technical report, Universität Leipzig.

[Strass and Wallner, 2014] Strass, H. and Wallner, J. P. (2014). Analyzing the Computational Complexity of Abstract Dialectical Frameworks via Approximation Fixpoint Theory. In *Proceedings of the Fourteenth International Conference on Principles of Knowledge Representation and Reasoning (KR-2014)*.

[Thang et al., 2009] Thang, P. M., Dung, P. M., and Hung, N. D. (2009). Towards a Common Framework for Dialectical Proof Procedures in Abstract Argumentation. *Journal of Logic and Computation*, 19(6):1071–1109.

[Toni and Sergot, 2011] Toni, F. and Sergot, M. (2011). Argumentation and Answer Set Programming. In Balduccini, M. and Son, T. C., editors, *Logic Programming, Knowledge Representation, and Nonmonotonic Reasoning: Essays in Honor of Michael Gelfond*, volume 6565 of *Lecture Notes in Computer Science*, pages 164–180. Springer.

[Toulmin, 2003] Toulmin, S. (2003). *The Uses of Argument.* Cambridge University Press.

[Tseitin, 1968] Tseitin, G. S. (1968). On the Complexity of Derivations in the Propositional Calculus. *Studies in Mathematics and Mathematical Logic*, Part II:115–125.

[Tullio and Grasso, 2011] Tullio, E. D. and Grasso, F. (2011). A Model for a Motivational System Grounded on Value Based Abstract Argumentation Frameworks. In Kostkova, P., Szomszor, M., and Fowler, D., editors, *Proceedings of the Fourth International Conference on Electronic Healthcare (eHealth-2011)*, volume 91, pages 43–50. Springer.

[Turing, 1936] Turing, A. M. (1936). On Computable Numbers, with an Application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 42:230–265.

[Verheij, 2007] Verheij, B. (2007). A Labeling Approach to the Computation of Credulous Acceptance in Argumentation. In Veloso, M. M., editor, *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI-2007)*, pages 623–628.

[Wakaki and Nitta, 2008] Wakaki, T. and Nitta, K. (2008). Computing Argumentation Semantics in Answer Set Programming. In *New Frontiers in Artificial Intelligence, JSAI 2008 Conference and Workshops, Revised Selected Papers*, volume 5447 of *Lecture Notes in Computer Science*, pages 254–269.

[Wallner et al., 2013] Wallner, J. P., Weissenbacher, G., and Woltran, S. (2013). Advanced SAT Techniques for Abstract Argumentation. In Leite, J., Son, T. C., Torroni, P., van der Torre, L., and Woltran, S., editors, *Proceedings of the Fourteenth International Workshop on Computational Logic in Multi-Agent Systems (CLIMA-2013)*, volume 8143 of *Lecture Notes in Computer Science*, pages 138–154. Springer.

[Woltran, 2003] Woltran, S. (2003). *Quantified Boolean Formulas: Theory and Practice*. PhD thesis, Technische Universität Wien, Institut für Informationssysteme.

[Zhang, 2009] Zhang, H. (2009). Combinatorial Designs by SAT Solvers. In Biere, A., Heule, M., van Maaren, H., and Walsh, T., editors, *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, pages 533–568. IOS Press.

[Zhang, 2005] Zhang, L. (2005). On Subsumption Removal and On-the-Fly CNF Simplification. In Bacchus, F. and Walsh, T., editors, *Proceedings of the Eighth International Conference on Theory and Applications of Satisfiability Testing (SAT-2005)*, volume 3569 of *Lecture Notes in Computer Science*, pages 482–489. Springer.

[Zhang and Malik, 2002a] Zhang, L. and Malik, S. (2002a). Conflict Driven Learning in a Quantified Boolean Satisfiability Solver. In Pileggi, L. T. and Kuehlmann, A., editors, *Proceedings of the IEEE/ACM International Conference on Computer-aided Design (ICCAD-2002)*, pages 442–449. ACM.

[Zhang and Malik, 2002b] Zhang, L. and Malik, S. (2002b). Towards a Symmetric Treatment of Satisfaction and Conflicts in Quantified Boolean Formula Evaluation. In Hentenryck, P. V., editor, *Proceedings of the Eighth International Conference on Principles and Practice of Constraint Programming - (CP-2002)*, volume 2470 of *Lecture Notes in Computer Science*, pages 200–215. Springer.