Reasoning in Abstract Dialectical Frameworks Using Quantified Boolean Formulas[1]

Martin DILLER, Johannes Peter WALLNER and Stefan WOLTRAN
Institute of Information Systems, Vienna University of Technology, Austria

Abstract Abstract dialectical frameworks (ADFs) constitute a recent and powerful generalization of Dung's argumentation frameworks (AFs), where the relationship between the arguments is specified via Boolean formulas. Recent results have shown that this enhancement comes with the price of higher complexity compared to AFs. In fact, acceptance problems in the world of ADFs can be hard even for the third level of the polynomial hierarchy. In order to implement reasoning problems on ADFs, systems for quantified Boolean formulas (QBFs) thus are suitable engines to be employed. In this paper we present QBF encodings on ADF problems generalizing recent work on QBFs for AF labellings. Our encodings not only provide a uniform and modular way of translating reasoning in ADFs to QBFs, but also build the basis for a novel system. We present a prototype implementation for the admissible and preferred semantics and evaluate its performance in comparison with another state-of-the-art tool for ADFs.

Keywords. Abstract dialectical frameworks, systems, quantified Boolean formulas

1. Introduction

Since its invention by Dung [1], abstract argumentation constitutes one of the main research branches for computational models of argument. Hereby, the conflict resolution between arguments is done by abstracting away from the arguments' contents, yielding a simple yet powerful framework for reasoning used as a core model in many argumentation tasks and applications. Although very general, Dung's argumentation frameworks (AFs) do not directly support modelling more complex interactions between arguments. For this and similar issues, several extensions of AFs have been proposed to date (e.g. those presented in [2,3,4]), one of the most general being that of abstract dialectical frameworks (ADFs) [5]. In this last framework, acceptance conditions in the form of arbitrary Boolean formulas are associated to every argument. Proper generalizations of Dung's semantics have been given in [6].

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 for AFs, most of the reasoning

[1]This research has been supported by the Austrian Science Fund (FWF) through projects I1102 and W1255-N23.

tasks have been shown to suffer from high computational complexity [7]. For ADFs, the complexity of many of the main reasoning tasks with respect to the generalized semantics jump one level of the polynomial hierarchy [8], resulting in some of the reasoning tasks even being on the third level of the polynomial hierarchy.

To deal with this high complexity, naive implementations for ADFs are not promising. Instead, one should aim at reducing the tasks in question to some other suitable language for which sophisticated systems already exist. In this work we follow this idea and present translations of some of the main reasoning tasks defined for ADFs into the satisfiability problem of quantified Boolean formulas (QBFs). QBFs are an extension of propositional logic, allowing for quantification over propositional atoms. Its associated satisfiability problem (QSAT) is complete for PSPACE while types of QBFs classified according to the structure of quantification provide us with complete problems for each level of the polynomial hierarchy. These results indicate that reasoning problems for ADFs can be efficiently transformed into QSAT problems with a certain quantifier structure. Employing the framework of QBFs thus allows for uniform translations of ADF reasoning tasks. Moreover, these translations can be done in a manner that is “sensitive” to the inherent complexity of the reasoning tasks. Finally, increasingly efficient solvers for QSAT (see, for example, the results presented in [9]) can be used for practical realizations.

On the theoretical side, our work continues the line of of study that has been initiated for Dung style argumentation frameworks in [10] and [11]. In both of these works, reductions of the problems of evaluating Dung style AFs 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 presents some parallels with the approach followed in [6] to define the semantics for ADFs. Thus this work has been particularly influential for the approach we follow to give encodings for the reasoning problems defined for ADFs.

Since we also present and give a first assessment of the performance of a prototype system based on the encodings presented in this work, from a practical perspective our work can be placed among “reduction-based approaches” [12] to implementation of argumentations systems. In particular, we give some first hints as to the potential benefits of a QBF based implementation approach for reasoning on ADFs in regards to the system `DIAMOND` [13], which is the only other comparable ADF system known to us and is based on reductions to the language of Answer-Set Programming (ASP).

We briefly summarize the main contributions of our work:

1. We provide “complexity-sensitive” encodings of the main reasoning problems defined for ADFs (evaluation, credulous acceptance, skeptical acceptance) with respect to some of the major semantics (two valued models, admissible, complete, preferred, grounded, stable) in the context of QBFs.
2. We present a prototype software system for reasoning on ADFs based on the encodings we provide in this work and report on preliminary experiments comparing our prototype system with the ADF system `DIAMOND`.

2. Background

2.1. Quantified Boolean Formulas

We recall the necessary background of Boolean logic and quantified Boolean formulas (QBFs) [14]. Quantified Boolean logic is an extension to propositional logic.

The basis of propositional logic is a set of propositional variables $\mathscr{P}$, to which we also refer to as atoms. Propositional formulas are built as usual from the connectives $\wedge, \vee, \rightarrow, \leftrightarrow$ and $\neg$, denoting the logical conjunction, disjunction, (material) implication, equivalence and negation respectively. As for truth constants, we use $\top$ for the value true and $\bot := \neg\top$ for false. QBFs additionally use the universal quantifier $\forall$ and the existential quantifier $\exists$. If φ is a (quantified) Boolean formula, then $Qp\varphi$ is a QBF, with $Q \in \{\forall, \exists\}$ and $p \in \mathscr{P}$. QBFs may be nested using the logical connectives. Further $Q\{p_1, \ldots, p_n\}\varphi$ is a shortcut for $Qp_1 \cdots Qp_n\varphi$. The order of variables in consecutive quantifiers of the same type does not matter. We define shorthands for implication and equivalence: $\varphi \rightarrow \varphi' := \neg\varphi \vee \varphi'$ and $\varphi \leftrightarrow \varphi' := (\varphi \rightarrow \varphi') \wedge (\varphi' \rightarrow \varphi)$.

A propositional variable p in a QBF φ is free if it does not occur within the scope of a quantifier Qp and bound otherwise. If φ contains no free variable, then φ is said to be closed and otherwise open. Further we will write $\varphi[p/\psi]$ to denote the result of uniformly substituting each free occurrence of p with ψ in the formula φ.

An interpretation $\hat{v} : \mathscr{P} \rightarrow \{\mathbf{t}, \mathbf{f}\}$ defines for each propositional variable a truth assignment. We sometimes explicitly highlight that a $\hat{v}$ is defined on a set $X \subseteq \mathscr{P}$. The evaluation on atoms generalizes as usual to arbitrary formulas: Given a formula φ and an interpretation $\hat{v}$, then φ evaluates to true under $\hat{v}$ ($\hat{v}$ satisfies φ or $\hat{v}$ is a model of φ, denoted by $\hat{v} \models \varphi$) if one of the following holds, with $p \in \mathscr{P}$ and QBFs ψ, ψ_1 and ψ_2.

- $\varphi = p$ and $\hat{v}(p) = \mathbf{t}$;
- $\varphi = \top$;
- $\varphi = \neg\psi$ and ψ does not evaluate to true under $\hat{v}$;
- $\varphi = \psi_1 \wedge \psi_2$ and both ψ_1 and ψ_2 evaluate to true under $\hat{v}$;
- $\varphi = \psi_1 \vee \psi_2$ and one of ψ_1 and ψ_2 evaluates to true under $\hat{v}$;
- $\varphi = \exists p\psi$ and one of $\psi[p/\top]$ and $\psi[p/\bot]$ evaluates to true under $\hat{v}$;
- $\varphi = \forall p\psi$ and both $\psi[p/\top]$ and $\psi[p/\bot]$ evaluate to true under $\hat{v}$.

We extend the evaluation function $\hat{v}$ to formulas, i.e. $\hat{v}(\varphi)$ is the evaluation of φ under $\hat{v}$.

Normal forms of (quantified) formulas play an important role for theoretical and practical purposes. For unquantified formulas a well-known normal form is the conjunctive normal form (CNF). A formula is in CNF if it is a conjunction of disjunctions of literals. A literal is either an atom or a negated atom. Translations of arbitrary Boolean formulas to CNF are possible in polynomial time via a transformation given by Tseitin [15].

A well-known normal form for QBFs is the prenex normal form (PNF). A QBF φ is in PNF if φ is of the form $Q_1, \ldots, Q_n\psi$ with $Q_i \in \{\forall, \exists\}$ for $1 \leq i \leq n$ and ψ a quantifier-free formula. If ψ is in CNF, then φ is in prenex conjunctive normal form (PCNF). The computational complexity of deciding whether φ is satisfiable depends on the prefix type. Every propositional formula has the prefix type $\Sigma_0 = \Pi_0$. Let φ be a closed QBF with prefix type Σ_i (respectively, Π_i) and X a set of $m > 0$ propositional variables. Then the formula $\forall X\varphi$ (respectively $\exists X\varphi$) is of type Π_{i+1} (respectively Σ_{i+1}) for $i \geq 0$.

We assume familiarity with the complexity classes P, NP and coNP. We also make use of the polynomial hierarchy, that can be defined (using oracle Turing machines as superscript) as follows: $\Sigma_0^P = \Pi_0^P = \mathrm{P}$, $\Sigma_{i+1}^P = \mathrm{NP}^{\Sigma_i^P}$, $\Pi_{i+1}^P = \mathrm{coNP}^{\Sigma_i^P}$ for $i \geq 0$. In general, deciding whether φ is satisfiable is Σ_k^P complete if φ has prefix type Σ_k and otherwise if φ has prefix type Π_k, then the problem is Π_k^P complete ($k \geq 1$).

2.2. Abstract Dialectical Frameworks

An abstract dialectical framework (ADF) is a directed graph whose vertices represent statements which can be accepted or not. The links represent dependencies: the status of a node s only depends on the status of its parents (denoted $\mathit{parents}(s)$), that is, the nodes with a direct link to s. In addition, each node s has an associated acceptance condition C_s specifying the conditions under which s can be accepted. C_s is a function assigning to each subset of $\mathit{parents}(s)$ one of the truth values $\mathbf{t}$, $\mathbf{f}$. Intuitively, if for some $R \subseteq \mathit{parents}(s)$ we have $C_s(R) = \mathbf{t}$, then s can be accepted provided the nodes in R are accepted and those in $\mathit{parents}(s) \setminus R$ are not accepted.

Definition 1. *An* abstract dialectical framework *is a tuple $D = (S, L, C)$ where*

- *S is a set of statements (positions, nodes),*
- *$L \subseteq S \times S$ is a set of links,*
- *$C = \{C_s\}_{s \in S}$ is a set of total functions $C_s : 2^{\mathit{parents}(s)} \to \{\mathbf{t}, \mathbf{f}\}$, one for each statement s. C_s is called acceptance condition of s.*

In many cases it is convenient to represent the acceptance conditions C as a collection $\{\varphi_s\}_{s \in S}$ of propositional formulas. This leads to the logical representation of ADFs we use in this paper where an ADF D is a pair (S, C) with the set of links L implicitly given as $(a, b) \in L$ iff a appears in φ_b.

We consider ADF semantics as defined in [6]. A semantics σ assigns to an ADF D a collection of two- or three-valued interpretations, denoted by $\sigma(D)$. To distinguish between interpretations of ADFs and QBFs we use $\hat{v}, \hat{w}$ for QBF interpretations and v, w for interpretations of ADFs. For σ we consider in this paper admissible, complete, grounded, preferred, two-valued and stable semantics of ADFs. We denote the semantics by *adm*, *com*, *grd*, *prf*, *mod* and *stb*.

The interpretations map statements to truth values. We use $\{\mathbf{t}, \mathbf{f}, \mathbf{u}\}$ as truth values, denoting true, false and undecided respectively. The three truth values are partially ordered by $\leq_i$ according to their information content: we have $\mathbf{u} <_i \mathbf{t}$ and $\mathbf{u} <_i \mathbf{f}$ and no other pair in $<_i$. The information ordering $\leq_i$ extends in a straightforward way to interpretations v_1, v_2 over S in that $v_1 \leq_i v_2$ iff $v_1(s) \leq_i v_2(s)$ for all $s \in S$. A three-valued interpretation v is two-valued if all statements are mapped to either true or false. For a three-valued interpretation v, we say that a two-valued interpretation w extends v iff $v \leq_i w$. This means that all statements mapped to $\mathbf{u}$ by v are mapped to $\mathbf{t}$ or $\mathbf{f}$ by w. We denote by $[v]_2$ the set of all two-valued interpretations that extend v.

For an ADF $D = (S, C)$, $s \in S$ and a three-valued interpretation v, the characteristic function $\Gamma_D(v) = v'$ is given by

$$v'(s) = \begin{cases} \mathbf{t} \text{ if } w(\varphi_s) = \mathbf{t} \text{ for all } w \in [v]_2 \\ \mathbf{f} \text{ if } w(\varphi_s) = \mathbf{f} \text{ for all } w \in [v]_2 \\ \mathbf{u} \text{ otherwise} \end{cases}$$

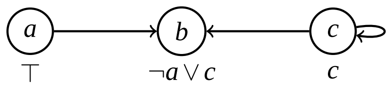


Figure 1.: ADF example

Table 1.: Complexity results for semantics of ADFs.

σ	*adm*	*com*	*prf*	*grd*	*mod*	*stb*
Cred_σ	Σ_2^P-c	Σ_2^P-c	Σ_2^P-c	coNP-c	NP-c	Σ_2^P-c
Skept_σ	trivial	coNP-c	Π_3^P-c	coNP-c	coNP-c	Π_2^P-c

That is, the operator returns a three-valued interpretation, mapping a statement s to $\mathbf{t}$, or respectively $\mathbf{f}$, if all two-valued interpretations extending v evaluate φ_s to true, respectively false. If there are $w_1, w_2 \in [v]_2$, s.t. $w_1(\varphi_s) = \mathbf{t}$ and $w_2(\varphi_s) = \mathbf{f}$, then the result is $\mathbf{u}$. Note that the characteristic function is defined on three-valued interpretations, while we evaluate acceptance conditions under two-valued interpretations (two-valued extensions of three-valued interpretations).

Definition 2. *Let $D = (S, L, C)$ be an ADF. A three-valued interpretation v is*

- *in $adm(D)$ if $v \leq_i \Gamma_D(v)$;*
- *in $com(D)$ if $v = \Gamma_D(v)$;*
- *in $grd(D)$ if $v \in com(D)$ and there is no $w \in com(D)$ with $w <_i v$;*
- *in $prf(D)$ if $v \in adm(D)$ and there is no $w \in adm(D)$ with $v <_i w$.*

For any ADF we have that all preferred interpretations are complete and all complete interpretations are admissible [6]. The *mod* and *stb* semantics rely on two-valued interpretations. A two-valued interpretation v is a model of an ADF $D = (S, C)$ if $v(s) = v(\varphi_s)$ for all $s \in S$. Stable models are defined via a reduct.

Definition 3. *Let $D = (S, L, C)$ be an ADF with $C = \{\varphi_s\}_{s \in S}$. A two-valued model v of D is a stable model of D iff $E_v = \{s \in S \mid v(s) = \mathbf{t}\}$ equals the statements set to true in the grounded interpretation of the reduced ADF $D^v = (E_v, L^v, C^v)$, where $L^v = L \cap (E_v \times E_v)$ and for $s \in E_v$ we set $\varphi_s^v = \varphi_s[b/\mathbf{f} : v(b) = \mathbf{f}]$.*

In Figure 1 we see an example ADF $D = (\{a, b, c\}, C)$. The acceptance conditions are shown below the statements. This ADF has three complete interpretations: v_1, v_2 and v_3 with $v_1(a) = \mathbf{t}$, $v_1(b) = v_1(c) = \mathbf{u}$, $v_2(a) = v_2(b) = v_2(c) = \mathbf{t}$ and $v_3(a) = \mathbf{t}$, $v_3(b) = v_3(c) = \mathbf{f}$. Further v_1 is the grounded interpretation of D, both v_2 and v_3 are preferred interpretations and two valued models of D. Only v_3 is a stable model of D.

We recall two important reasoning tasks on ADFs for a semantics σ. A statement s is credulously accepted in an ADF D for σ if s is true in at least one $v \in \sigma(D)$. The corresponding decision problem is denoted by Cred_σ. If s is true in all $v \in \sigma(D)$, then it is skeptically accepted in D w.r.t. σ, the decision problem denoted by Skept_σ. The computational complexity of reasoning in ADFs is summarized in Table 1. The results were shown in [6,8].

3. Encodings

In this section we present our encodings of reasoning tasks for ADFs in QBFs. We develop for an ADF (S,C) and a semantics σ a *defining encoding*, denoted by $\mathscr{E}_\sigma[S,C]$. $\mathscr{E}_\sigma[S,C]$ is a QBF whose models correspond to the σ interpretations of the ADF (S,C).

We begin with some preliminaries. For ease of presentation we will slightly abuse our notation and use for an ADF $D=(S,C)$ the statements in S also as propositional variables. To encode expressions about three valued interpretations as QBFs we follow the procedure that has already been used in [11] to encode expressions about (three valued) labellings for AFs. Specifically, we assume that for every set S of propositional variables representing statements of some ADF the alphabet of QBFs also contains the set $S_3 := \{s^\oplus \mid s \in S\} \cup \{s^\ominus \mid s \in S\}$ of signed variables. The intended meaning of $s^\oplus$ is that s is accepted and $s^\ominus$ that s is rejected under some interpretation. Intuitively, a statement is undecided if both $s^\oplus$ and $s^\ominus$ are false in a model.

Since the definitions of semantics of ADFs often refer to several interpretations (e.g. the extensions of a given interpretation) we also use disjoint sets of propositional variables to implicitly refer to these different interpretations in our encodings. We distinguish the different sets by "priming" the variables in S, i.e. $S' := \{s' \mid s \in S\}$. Priming an already primed set adds another prime, e.g. $(S')' = \{s'' \mid s \in S\}$. The renaming via primes is straightforwardly extended to formulas and sets of formulas, i.e. φ' is obtained by replacing all variables s in φ by s'. Similarly for an ADF $D=(S,C)$ we define $C' := \{\varphi'_s\}_{s\in S}$. If we apply both signing and priming, then priming takes precedence, i.e. $S'_3 = \{s'^\oplus \mid s \in S\} \cup \{s'^\ominus \mid s \in S\}$.

ADF semantics are based on three or two truth values. Since there are four possible truth value assignments for a statement s via $s^\oplus, s^\ominus \in S_3$, we need to restrict attention to *coherent* interpretations for QBFs, which exclude the possibility for a model to satisfy both $s^\oplus$ and $s^\ominus$.

Definition 4. *Let S be a set. A two valued interpretation $\hat{v}$ is* coherent *on S_3 if there is no $s \in S$ such that $\hat{v}(s^\oplus) = \mathbf{t}$ and $\hat{v}(s^\ominus) = \mathbf{t}$.*

We now formally define the correspondence we require of the models $\hat{v} \models \mathscr{E}_\sigma[S,C]$ and $v \in \sigma(D)$ for an ADF $D=(S,C)$ and a semantics σ.

Definition 5. *Let $D=(S,C)$ be an ADF. We define two concepts of correspondences. We say that a two valued interpretation $\hat{v}$ on S corresponds to a* two valued *interpretation v on S, denoted as $\hat{v} \cong_S v$ if $v(s) = \hat{v}(s)$ for all $s \in S$. A coherent two valued interpretation $\hat{v}'$ on S_3 corresponds to a* three valued *interpretation v' on S, denoted as $\hat{v}' \cong_{S_3} v'$ if the following three conditions are met:*

a) $v'(s) = \mathbf{t}$ if and only if $\hat{v}'(s^\oplus) = \mathbf{t}$ and $\hat{v}'(s^\ominus) = \mathbf{f}$;
b) $v'(s) = \mathbf{f}$ if and only if $\hat{v}'(s^\oplus) = \mathbf{f}$ and $\hat{v}'(s^\ominus) = \mathbf{t}$; and
c) $v'(s) = \mathbf{u}$ if and only if $\hat{v}'(s^\oplus) = \mathbf{f}$ and $\hat{v}'(s^\ominus) = \mathbf{f}$.

That is, if $\hat{v} \cong_S v$, then $\hat{v}$ encodes a two valued interpretation v and we directly compare variables in S. Otherwise if $\hat{v}' \cong_{S_3} v'$, then $\hat{v}'$ encodes a three valued interpretation v on S and the correspondence is formally over the atoms in S_3. We extend these notions to sets of models of QBFs and interpretations of ADFs as follows. Let $\mathscr{V}$ be a set of QBF models, $\mathscr{W}$ be a set of (two or three valued) interpretations and $X \in \{S, S_3\}$. Then

$\mathscr{V} \cong_X \mathscr{W}$ iff (i) for each $\hat{v} \in \mathscr{V}$ there is a $w \in \mathscr{W}$, s.t. $\hat{v} \cong_X w$ and (ii) for each $w \in \mathscr{W}$ there is a $\hat{v} \in \mathscr{V}$, s.t. $\hat{v} \cong_X w$.

The encodings we present are all constructed following the same general strategy. For an ADF $D = (S,C)$ we define a QBF $\mathscr{E}_\sigma[S,C]$ with free variables in S_3 for $\sigma \in \{adm, com, prf, grd\}$ and S for $\sigma \in \{mod, stb\}$ such that $v \in \sigma(D)$ if and only if $\hat{v} \models \mathscr{E}_\sigma[S,C]$ for any $\hat{v}$ such that $\hat{v}$ corresponds to v. Thus $\mathscr{E}_\sigma[S,C]$ encodes the enumeration problem for ADFs with respect to σ into the model enumeration problem for QBFs.

When encoding the reasoning tasks as QBFs we make use of some simple modules. In the first place, given a set of statements S, the following formula "filters out" interpretations which are not coherent on S_3:

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

In order to encode the definitions of ADF semantics as QBFs we often need to express that $v(s) \leq_i v'(s)$ on all $s \in S$ for two interpretations of an ADF $D = (S,C)$. The formula

$$S_3 \leq_i S'_3 := \bigwedge_{s \in S}((s^\oplus \to s'^\oplus) \wedge (s^\ominus \to s'^\ominus))$$

does precisely that assuming both v and v' are three valued and the former is defined on S_3 and the latter on S'_3. The formula below

$$S_3 \leq_i S := \bigwedge_{s \in S}((s^\oplus \to s) \wedge (s^\ominus \to \neg s))$$

expresses this in case v is three valued but v' is two valued. In other words if $\hat{v} \models S_3 \leq_i S$ (and $\hat{v}$ is coherent on S_3) then $\hat{v}$ encodes two interpretations on S: a three valued interpretation v and a two valued interpretation v', s.t. $\hat{v} \cong_{S_3} v$, $\hat{v} \cong_S v'$ and $v' \in [v]_2$.

To encode admissible semantics of ADFs we make use of the following proposition.

Proposition 1. *Let $D = (S, \{\varphi_s\}_{s \in S})$ be an ADF and v a three valued interpretation on S. Then $v \in adm(D)$ iff $v(s) \leq_i w(\varphi_s)$ for all $s \in S$ and $w \in [v]_2$.*

This leads us to our first encoding for admissible semantics of ADFs. The following formula consists of two conjuncts for an ADF $D = (S,C)$. The left one states that each model $\hat{v} \models \mathscr{E}_{adm}[S,C]$ on the free variables S_3 is coherent. The right conjunct encodes the preceding proposition. Simply put, for the three valued interpretation v corresponding to $\hat{v}$ ($\hat{v} \cong_{S_3} v$) we have that for all the two valued extensions w of v it holds that $v(s) \leq_i w(\varphi_s)$.

$$\mathscr{E}_{adm}[S,C] := coh[S] \wedge \forall S\big((S_3 \leq_i S) \to \bigwedge_{s \in S}((s^\oplus \to \varphi_s) \wedge (s^\ominus \to \neg\varphi_s))\big)$$

We encode the enumeration of all complete interpretations of an ADF as follows.

$$\mathscr{E}_{com}[S,C] := \mathscr{E}_{adm}[S,C] \wedge \bigwedge_{s \in S}\big((\neg s^\oplus \wedge \neg s^\ominus) \to \exists S' \cup S''(S_3 \leq_i S' \wedge S_3 \leq_i S'' \wedge \varphi'_s \wedge \neg\varphi''_s)\big)$$

That is, the free variables of $\mathscr{E}_{com}[S,C]$ are S_3 and the first conjunct ensures that these correspond to an admissible interpretation. The right conjunct is slightly more involved.

If a statement is undecided (both $s^{\oplus}$ and $s^{\ominus}$ evaluate to false), then there must be two two valued extensions of the admissible interpretation, one evaluating the acceptance condition of s to true and another, which evaluates φ_s to false.

Using again the encoding for admissibility we encode preferred semantics in the next formula. The formula specifies that the result should correspond to an admissible interpretation v and for any admissible interpretation v' with greater or equal information content we require that $v(s) = v'(s)$ for any $s \in S$.

$$\mathscr{E}_{prf}[S,C] := \mathscr{E}_{adm}[S,C] \wedge \forall S'_3 \big(\mathscr{E}_{adm}[S',C'] \to (S_3 \leq_i S'_3 \to S'_3 \leq_i S_3)\big)$$

The grounded semantics uses a similar technique as we utilized for preferred. Instead of the module for admissible interpretations we use complete semantics and require that the result is information minimal. A three valued interpretation v is preferred for an ADF D if for all $v' \in adm(D)$ we have either that v and v' are incomparable w.r.t. $\leq_i$, or that $v' \leq_i v$. The grounded interpretation is the $\leq_i$-minimal complete interpretation and thus all complete interpretations are comparable to the grounded interpretation. Therefore the encoding for the grounded semantics is slightly simpler than the one for preferred, where the check for comparability has to be included.

$$\mathscr{E}_{grd}[S,C] := \mathscr{E}_{com}[S,C] \wedge \forall S'_3 (\mathscr{E}_{com}[S',C'] \to S_3 \leq_i S'_3)$$

Having established the encodings for the three valued semantics on ADFs we now proceed to the remaining two valued semantics. We begin with two valued models.

$$\mathscr{E}_{mod}[S,C] := \bigwedge_{s \in S} (s \leftrightarrow \varphi_s)$$

The last semantics we encode are the stable models of ADFs. We first require that the result corresponds to a two valued ADF model (first conjunct below). The two remaining conjuncts encode the computation via the reduct. Here we use a simple "trick". Intuitively, by conjoining the acceptance condition by s and thus inserting a statement of a different vocabulary than the ones in φ'_s, we achieve that if s is evaluated to false by a $\hat{v} \models \mathscr{E}_{stb}[S,C]$, then the acceptance condition can be seen as equivalent to $\bot$. This simulates the replacement of $\bot$ by a statement mapped to false in the reduct for the stable semantics, since the grounded interpretation sets all statements s to false, whose φ_s is equivalent to $\bot$. Otherwise if $\hat{v}(s) = \mathbf{t}$, then the acceptance condition is unchanged. The grounded module "computes" the grounded interpretation v of the reduct. We then require that a statement s' is set to true by v (i.e. $s'^{\oplus}$ is evaluated to true) iff s is set to true by $\hat{v}$. That is, the statements set to true in the two valued model are also true in the grounded interpretation of the reduct.

$$\mathscr{E}_{stb}[S,C] := \mathscr{E}_{mod}[S,C] \wedge \bigwedge_{s \in S} (s \leftrightarrow s'^{\oplus}) \wedge \mathscr{E}_{grd}[S', \{\varphi'_s \wedge s\}_{s \in S}]$$

The correctness of our encodings is summarized in the next proposition. Due to space limitations we have to omit the proof.

Proposition 2. *Let $D = (S,C)$ be an ADF, $\sigma \in \{adm, com, prf, grd\}$ and $\sigma' \in \{mod, stb\}$. It holds that $\{\hat{v} \models \mathscr{E}_\sigma[S,C]\} \cong_{S_3} \{v \in \sigma(D)\}$ and $\{\hat{v} \models \mathscr{E}_{\sigma'}[S,C]\} \cong_S \{v \in \sigma'(D)\}$.*

Having the defining encoding function for a semantics σ, encodings for the reasoning tasks we consider with respect to σ can be given in a standard fashion:

Proposition 3. *Let $D = (S,C)$ be an ADF, $s \in S$, $\sigma \in \{adm, com, prf, grd, mod, stb\}$. If $\sigma \in \{adm, com, prf, grd\}$, then let $x = s^{\oplus}$ and $T = S_3$. Otherwise if $\sigma \in \{mod, stb\}$, then let $x = s$ and $T = S$. It holds that*

- $\mathsf{Cred}_{\sigma}(s,D) = yes$ *iff* $\exists T\,(\mathscr{E}_{\sigma}[S,C] \wedge x)$ *is satisfiable; and*
- $\mathsf{Skept}_{\sigma}(S,D) = yes$ *iff* $\forall T\,(\mathscr{E}_{\sigma}[S,C] \rightarrow x)$ *is satisfiable.*

Many of these encodings not only provide a uniform way of deciding credulous or skeptical queries, but are also adequate w.r.t. their complexity. Since for any ADF $D = (S,C)$ and $s \in S$ we have $\mathsf{Cred}_{adm}(s,D) = \mathsf{Cred}_{com}(s,D) = \mathsf{Cred}_{prf}(s,D)$ we can use the encodings for admissibility in these cases. Transforming $\exists T\,(\mathscr{E}_{adm}[S,C] \wedge x)$ to PNF results in a formula of prefix type Σ_2, i.e. a QBF in a class of QBFs for which satisfiability is Σ_2^P complete (Cred_{adm} is also Σ_2^P complete). We similarly arrive at QBFs in PNF which match the corresponding complexity of the decision problems for Skept_{prf} and credulous as well as skeptical reasoning for two valued model semantics. For instance, the QBF $\forall T\,(\mathscr{E}_{prf}[S,C] \rightarrow x)$ can be transformed to a PNF with prefix type Π_3. Skeptical acceptance w.r.t. admissible semantics is trivial.

For the remaining queries our encodings do not result in QBFs with a prefix type matching the complexity of the corresponding decision problem on ADFs. The reason for this is that our encoding for grounded semantics, if translated to PNF, has a higher prefix type than the complexity of the corresponding decision problems would suggest. However, complexity adequate encodings for grounded semantics are possible [16]. These are more involved and incorporate more propositional variables per statement. Finally since $\mathsf{Skept}_{com}(s,D) = \mathsf{Skept}_{grd}(s,D)$, also skeptical acceptance w.r.t. complete semantics can be encoded with a lower prefix type than presented here.

4. System

The encodings presented in the previous section can be used to implement a “reasoner” without much effort. We have implemented a prototype of such a system which we call `QADF`[2]. It is essentially a compiler which, when given a representation of an ADF and a reasoning task of choice, outputs the encoding of the task as a QBF and then makes use of a state-of-the-art QSAT solver to solve it.

`QADF` is itself a `Unix` script called from command line. For the input format a representation that is also used in previous existing systems for ADFs including `DIAMOND` [13] is adopted. Each statement `s` is encoded via the string `statement(s)`. The acceptance condition `F` of `s` is specified in prefix notation via `ac(s,F)`. For example the acceptance conditions of the ADF from Figure 1 are encoded as follows: `ac(a,c(v))`, `ac(b,or(neg(a),c))`, and `ac(c,c)`, where `c(v)` stands for $\top$.

The translation from ADFs to QBFs which we have implemented in Scala (2.9.3) generates its output in the QDIMACS format which is an extension of the DIMACS format assumed by most SAT solvers. The output formulas are generated using the Tseitin transformation (adapted for QBFs) optimized to our setting. As back-end we use the

[2] `http://www.dbai.tuwien.ac.at/proj/adf/qadf`

Table 2.: Number of time-outs of `DIAMOND` and `QADF` for computation of credulous and skeptical acceptance of statements with respect to admissible and preferred semantics. Instance sets are partitioned according to the number of statements the ADFs in them have.

(a) Credulous acceptance - admissible

Statements	`DIAMOND`	`QADF`
10	0/120	0/120
20	0/120	0/120
30	0/120	0/120
40	0/120	0/120
50	0/120	1/120
60	0/120	3/120

(b) Skeptical acceptance - preferred

Statements	`DIAMOND`	`QADF`
10	0/60	0/60
15	0/60	0/60
20	30/60	0/60
25	54/60	52/60
30	45/60	60/60

QSAT solver `DepQBF` [17] which was placed first in the main track of the QBFEVAL 2010 competition for QSAT solvers. We also use `Bloqqer` [18] for pre-processing.

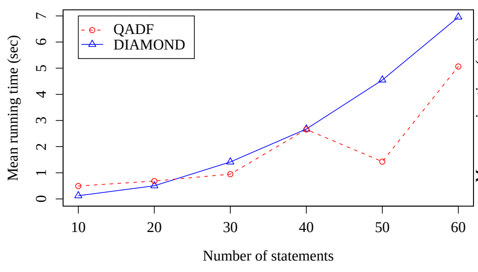
We have carried out preliminary experiments to determine the performance of `QADF` focusing on Cred_{adm} and Skept_{prf} and comparing it to that of `DIAMOND`. As already indicated, `DIAMOND` is based on encodings to answer set programming and uses the `Potassco` bundle of ASP tools.

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 0.2) we use the latest version of `DIAMOND` (version 0.9) which, in turn, requires `gringo` (version 3.0.4), `clasp` (version 2.1.5), and `claspD` (version 1.1.4) from `Potassco`. `DIAMOND` only does enumeration of interpretations out of the box so we adapted it (using ASP constraints) in order to carry out credulous and skeptical reasoning. For `QADF` we use `Bloqqer` (version 031) and `DepQBF` (version 2.0).

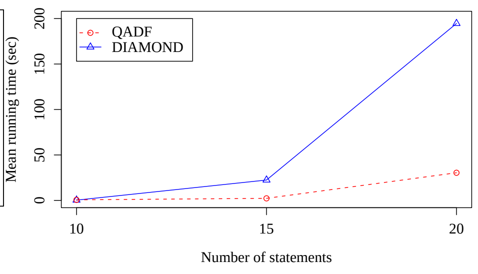
In order to generate random instances of ADFs for our experiments we have used the generator used to test the precursor of `DIAMOND`, `ADFSys` [19]. This generator aligns statements in a grid, with a width of 7 in our experiments. The total number of statements n is the parameter for the size of the generated ADF. Each statement in the grid has as its parents a subset of the eight neighbours in the grid. The acceptance condition for a statement s is created as follows with x one of s's eight neighbors in the grid. The probability that the link (x,s) is symmetric is 0.5 and otherwise with equal chance a link in one direction. Given the set of parents of s, we construct φ_s by iteratively connecting the parents via $\wedge$ or $\vee$. Additionally there is a 0.2 probability that a statement is replaced during this process by a truth constant (again with equal probability for $\top$ or $\bot$).

For the experiments for Cred_{adm} we have generated 240 ADF instances, with 40 instances of 10, 20, 30, 40, 50, and 60 statements. For Skept_{prf} we generated 100 ADF instances, with 20 instances of 10, 15, 20, 25, and 30 statements. For each ADF instance we also generated reasoning tasks for 3 arbitrarily chosen statements (the statements with identifiers 3, 5, and 7). This means that the total number of instances is 720 for Cred_{adm} and 300 for Skept_{prf}. 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 calculated via the Unix `time` utility.

In our experiments no time-outs occurred for `DIAMOND` for Cred_{adm}, while there were very few (4) for `QADF` on instances with 50 or more statements. There were a



(a) Credulous acceptance - admissible



(b) Skeptical acceptance - preferred

Figure 2.: Mean running times of DIAMOND and QADF for computation of credulous and skeptical acceptance of statements with respect to admissible and preferred semantics. Instance sets are partitioned according to the number of statements the ADFs in them have.

significant number of time-outs for Skept_{prf} for instances with 20 or more statements for DIAMOND and 25 or more statements for QADF. Table 2 shows the time-outs.

Figures 2a and 2b represent the mean running times of QADF and DIAMOND, where the instances with time-outs are disregarded. We have only included mean times for instance sets with ADFs with 10, 15, and 20 statements for Skept_{prf} given the significant number of time-outs for the instance sets with ADFs having more statements. Note that for ADFs with 20 statements DIAMOND times-out on 30 instances, while QADF on none.

The results of our preliminary experiment suggests an acceptable performance of QADF for Cred_{adm} on instances with up to at least 40 statements, while there are clearly performance issues for Skept_{prf} for ADF instances with already 20 or more statements. This situation reflects the relative intrinsic complexity of both tasks.

The results do not indicate any clear and significant difference between QADF and DIAMOND (adapted by us to support skeptical and credulous reasoning), except in the case of Skept_{prf} for ADFs with 20 statements where DIAMOND timed-out on half of the instances, while QADF was able to solve all these instances with lower mean computation time than DIAMOND on those instances in which it did not time-out. This case is somewhat curious considering that for ADFs with 25 or more statements both DIAMOND and QADF timed-out on most of the instances. For ADFs with 30 instances DIAMOND at least manages to solve 15 instances while QADF solves none in the allotted time.

The situation described does not warrant any other conclusion than that QADF can clearly compete with DIAMOND on the reasoning tasks studied by us so far and that further experiments are necessary to more clearly assess the comparative advantages and limits of both systems. 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 have a rather simple structure and additionally fall into the class of “bipolar ADFs” [5], which have slightly milder complexity [8].

5. Conclusion

We have presented encodings of several reasoning tasks defined on ADFs into the language of QBFs as well as a prototype system for ADFs based on these. This has enabled us to take advantage of recent developments for QBF-solvers and thereby tackle the high complexity of reasoning problems for ADFs. Preliminary experiments show a competitive performance to another existing ADF system, but compared to the success of QBF-solvers in other domains, the results indicate that there is room for improvement.

Thus, future work will focus on tuning of the encodings to improve performance. In particular, it may be that QBF-solvers are struggling to efficiently deal with the signed variables in our encodings, a concept we have borrowed from [11]; this issue needs further investigation. Another point on our agenda is to specialize our encodings for "less complex" ADFs (e.g. bipolar ADFs). Finally, additional complexity in the structure of the encodings due to the Tseitin transformation may be avoided by using a QSAT solver that does not require the input to be in PCNF.

References

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

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

[3] Cayrol, C., Lagasquie-Schiex, M.: On the Acceptability of Arguments in Bipolar Argumentation Frameworks. In: ECSQARU'05. Volume 3571 of LNCS. Springer (2005) 378–389

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

[5] Brewka, G., Woltran, S.: Abstract Dialectical Frameworks. In: KR'10. (2010) 102–111

[6] Brewka, G., Ellmauthaler, S., Strass, H., Wallner, J.P., Woltran, S.: Abstract Dialectical Frameworks Revisited. In: IJCAI'13. (2013) 803–809

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

[8] Strass, H., Wallner, J.P.: Analyzing the Computational Complexity of Abstract Dialectical Frameworks via Approximation Fixpoint Theory. In: KR'14. (2014) 101–110

[9] Lonsing, F., Seidl, M., eds.: Informal Workshop Report on the International Workshop on Quantified Boolean Formulas 2013. (2013)

[10] Egly, U., Woltran, S.: Reasoning in Argumentation Frameworks Using Quantified Boolean Formulas. In: COMMA'06. Volume 144 of FAIA. (2006) 133–144

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

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

[13] Ellmauthaler, S., Strass, H.: The DIAMOND System for Argumentation: Preliminary Report. In: ASPOCP'13. (2013) 97–108

[14] Büning, H.K., Bubeck, U.: Theory of Quantified Boolean Formulas. In: Handbook of Satisfiability. Volume 185 of FAIA. (2009) 735–760

[15] Tseitin, G.S.: On the Complexity of Derivations in the Propositional Calculus. Studies in Mathematics and Mathematical Logic **Part II** (1968) 115–125

[16] Diller, M.: Solving Reasoning Problems on Abstract Dialectical Frameworks via Quantified Boolean Formulas. Master's thesis, Technische Universität Wien, Institut für Informationssysteme (2014)

[17] Lonsing, F., Biere, A.: DepQBF: A Dependency-Aware QBF Solver. Journal on Satisfiability, Boolean Modelling and Computation **7** (2010) 71–76

[18] Biere, A., Lonsing, F., Seidl, M.: Blocked Clause Elimination for QBF. In: CADE'11. Volume 6803 of LNCS. (2011) 101–115

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