Relating the Semantics of Abstract Dialectical Frameworks and Standard AFs*

Gerd Brewka
Computer Science Institute
Univ. of Leipzig, Germany

Paul E. Dunne
Dept. of Computer Science
Univ. of Liverpool, UK

Stefan Woltran
Institute of Information Systems
Vienna Univ. of Technology, Austria

Abstract

One criticism often advanced against abstract argumentation frameworks (AFs), is that these consider only one form of interaction between atomic arguments: specifically that an argument *attacks* another. Attempts to broaden the class of relationships include bipolar frameworks, where arguments support others, and abstract dialectical frameworks (ADFs). The latter, allow "acceptance" of an argument, x, to be predicated on a given propositional function, C_x, dependent on the corresponding acceptance of its parents, i.e. those y for which $\langle y, x\rangle$ occurs. Although offering a richly expressive formalism subsuming both standard and bipolar AFs, an issue that arises with ADFs is whether this expressiveness is achieved in a manner that would be *infeasible* within standard AFs. Can the semantics used in ADFs be mapped to some AF semantics? How many arguments are needed in an AF to "simulate" an ADF? We show that (in a formally defined sense) any ADF can be simulated by an AF of similar size and that this translation can be realised by a polynomial time algorithm.

Keywords: Argumentation, Knowledge representation

1 Introduction

The widely studied approach to abstract argumentation proposed by Dung [1995] uses a directed graph scheme (called an *argumentation framework* – AF), $\langle \mathcal{X}, \mathcal{A}\rangle$, in which $\mathcal{X}$ defines a set of atomic *arguments* and $\mathcal{A}$ a binary relation between these. In Dung's model, $\mathcal{A}$ is seen as capturing a concept of "incompatibility" of arguments, x and y say, in the sense that $\langle x, y\rangle \in \mathcal{A}$ indicates "the argument x *attacks* the argument y". Recognising that interaction between arguments may involve relationships other than that of "attack", [Cayrol and Lagasquie-Schiex, 2005; Amgoud *et al.*, 2008] introduced *bipolar* frameworks wherein an additional *support* relation, $\mathcal{R}$, between arguments is specified: hence $\langle x, y\rangle \in \mathcal{R}$ allows both x and y simultaneously to be accepted on the grounds that "the argument x *provides support* for the argument y". Recently, Brewka and Woltran [2010] have proposed a new model – Abstract Dialectical Frameworks (ADFs) – in which such treatments of how arguments interact are generalised as follows. Each argument, x, is associated with an *acceptance condition*, C_x, which is some propositional function whose truth status is determined by the corresponding (values of) the acceptance conditions for those arguments, y, with $\langle y, x\rangle$ a link in the ADF. In this way standard Dung style AFs are recovered by setting as acceptance condition for each argument the function $\wedge_{y\,:\,\langle y,x\rangle}\,\neg y$, i.e. x is accepted if none of its parents is.[1] The concept of associating individual acceptance conditions with arguments provides ADFs with a rich expressive capability that appears to be a more intuitive and natural mechanism with which to describe application scenarios, e.g. in later work [Brewka and Gordon, 2010] it is shown that the Carneades formalism of [Gordon *et al.*, 2007] can be reconstructed using ADFs.

Although it is not hard to show that standard AFs may be directly realised as ADFs with the basic stable extension semantics of the former corresponding to the so-called *models* of the latter, it is far from clear to what extent efficient translations in the reverse direction are possible: naive simulations in which ADF models map to AF stable extensions lead to an exponential increase in the number of arguments. The main result of this paper is to show that any ADF may be simulated (in a sense to be made precise) by an AF only polynomially larger and that this translation can be effected by a polynomial time algorithm.

We first review the basic elements of AFs and ADFs in Section 2; Section 3 recalls some supporting ideas from Boolean function complexity and its relation to AFs as these are applied in our constructions. The main results of the paper are presented in Sections 4 and 5 with discussion of these and conclusions offered in Section 6.[2]

2 Background

We first recall the abstract model of argumentation presented in [Dung, 1995].

*The third author is supported by the Vienna Science and Technology Fund (WWTF) under grant ICT08-028.

[1] An alternative mechanism is implicit in [Gabbay, 2009].

[2] Due to space restrictions some proofs were omitted. Proofs for results fundamental for the rest of the paper are included.

Definition 1 *An* argumentation framework *(*AF*) is a pair* $\langle \mathcal{X}, \mathcal{A} \rangle$*, in which* $\mathcal{X}$ *is a finite set of* arguments *and* $\mathcal{A} \subseteq \mathcal{X} \times \mathcal{X}$ *is the* attack relationship*. A pair* $\langle x, y \rangle \in \mathcal{A}$ *is referred to as '*y is attacked by x*' or '*x attacks y*'; *$x \in \mathcal{X}$ *is* acceptable with respect to $S \subseteq \mathcal{X}$ *if for every* $y \in \mathcal{X}$ *that attacks* x *there is some* $z \in S$ *that attacks* y*. The* characteristic function*,* $\mathcal{F} : 2^{\mathcal{X}} \rightarrow 2^{\mathcal{X}}$ *is the mapping which, given* $S \subseteq \mathcal{X}$*, reports the set of* $y \in \mathcal{X}$ *for which* y *is acceptable to* S*. The* grounded *extension is the (unique) least fixed point of* $\mathcal{F}$*, i.e. defining* $\mathcal{F}^0 = \emptyset$*,* $\mathcal{F}^k = \mathcal{F}(\mathcal{F}^{k-1})$ *the set corresponding to the least value* i *with* $\mathcal{F}^i = \mathcal{F}^{i-1}$*. We denote by* $GE(\langle \mathcal{X}, \mathcal{A} \rangle)$ *the arguments in the grounded extension of* $\langle \mathcal{X}, \mathcal{A} \rangle$*. The set* $S \subseteq \mathcal{X}$ *is* conflict free *if no argument in* S *is attacked by any other argument in* S*. A conflict free set* S *is:* admissible *if every* $y \in S$ *is acceptable w.r.t* S*; and a* stable extension *if every* $y \notin S$ *is attacked by some* $x \in S$*;*

The development of Dung's model which is the focus of the current paper are the *Abstract Dialectical Frameworks* (ADFs) from [Brewka and Woltran, 2010].

Definition 2 *An abstract dialectical framework is a triple* $\langle \mathcal{S}, \mathcal{L}, \mathcal{C} \rangle$ *with* $\mathcal{S} = \{s_1, \ldots, s_m\}$ *a finite set of* arguments*,* $\mathcal{L} \subseteq \mathcal{S} \times \mathcal{S}$ *a set of binary links, and* $\mathcal{C} = \langle C_1, C_2, \ldots, C_m \rangle$ *a collection of* acceptance conditions *described as (total)* propositional functions $C_i : 2^{par(s_i)} \rightarrow \langle \top, \bot \rangle$ *where* $par(s_i) = \{r \in \mathcal{S} : \langle r, s_i \rangle \in \mathcal{L}\}$*. A subset* M *of* $\mathcal{S}$ *is* conflict-free *if for all* $s \in M$*,* $C_s[M \cap par(s)] = \top$*;* M *is a* model *if* M *is conflict-free and for each* $s \in S$*,* $C_s[M \cap par(s)] = \top$ *implies* $s \in M$*, i.e.* M *is a model if for for each* $s \in S$*:* $s \in M$ *if and only if* $C_s[M \cap par(s)] = \top$*.*

In Section 5 we will recall the notions of well-founded and stable models for ADFs.

We use the following notational convention: for $par(s) = \langle r_1, \ldots, r_m \rangle$, $C_s[R]$ with $R \subseteq par(s)$ corresponds to the assignment α_R of propositional values in $\langle \bot, \top \rangle^m$ with $\alpha_i = \top$ if and only if $r_i \in R$. Similarly, for $\alpha \in \langle \bot, \top \rangle^m$, $R_\alpha \subseteq par(s)$ contains exactly $\{r_i : \alpha_i = \top\}$. In general, in describing evaluations of C_s, square brackets ([ ]) will be used for subsets of $par(s)$, and parentheses – ( ) – for propositional assignments, so that $C_s[R]$ is the same as $C_s(\alpha_R)$ and $C_s(\alpha)$ corresponds to $C_s[R_\alpha]$.

Central to the results of Section 4 is the concept of "an AF *simulating* an ADF".

Definition 3 *Let* $\langle \mathcal{S}, \mathcal{L}, \mathcal{C} \rangle$ *be an* ADF*. The* AF $\langle \mathcal{X}, \mathcal{A} \rangle$ simulates $\langle \mathcal{S}, \mathcal{L}, \mathcal{C} \rangle$ *if* all *of the following properties hold.*

S1. $\mathcal{S} \subseteq \mathcal{X}$*.*

S2. If $M \subseteq \mathcal{S}$ *is a model of* $\langle \mathcal{S}, \mathcal{L}, \mathcal{C} \rangle$ *there is a subset,* Y_M *of* $\mathcal{X}$*, for which* $M \cup Y_M$ *is a stable extension of* $\langle \mathcal{X}, \mathcal{A} \rangle$*.*

S3. If $T \subseteq \mathcal{X}$ *is a stable extension of* $\langle \mathcal{X}, \mathcal{A} \rangle$ *then* $T \cap \mathcal{S}$ *is a model of* $\langle \mathcal{S}, \mathcal{L}, \mathcal{C} \rangle$*.*

In informal terms, the AF $H^D = \langle \mathcal{X}, \mathcal{A} \rangle$ simulates the ADF $D = \langle \mathcal{S}, \mathcal{L}, \mathcal{C} \rangle$ if we can map models in the latter to stable extensions in the former and *vice versa*.

3 Boolean functions and AFs

The presentation of an ADF $\langle \mathcal{S}, \mathcal{L}, \mathcal{C} \rangle$ in a computational setting must describe two components: the directed graph structure $\langle \mathcal{S}, \mathcal{L} \rangle$; and the set of *propositional functions* (acceptance conditions) prescribed by $\mathcal{C}$. In considering notions of how "efficiently" an arbitrary ADF, $\langle \mathcal{S}, \mathcal{L}, \mathcal{C} \rangle$, can be simulated using an AF, $\langle \mathcal{X}, \mathcal{A} \rangle$, we cannot assess this *solely* in terms of how $|\mathcal{X}|$ relates to $|\mathcal{S}|$. Some cost is incurred in describing $\mathcal{C}$ and thus a more "reasonable" measure of simulation efficiency is in terms how $|\mathcal{X}|$ compares to the total cost of describing the set $|\mathcal{C}|$. In order to formalise this we need to consider what representations for propositional functions could be used together with associated size measures. There is, of course, a wide variety of schemes that have been considered for representing propositional functions. We shall concentrate on two related methods: propositional formulae over a fixed logical basis (an approach that has been widely exploited in AI contexts); and the Boolean combinational network model (or straight-line program) which is of importance in providing a benchmark for efficient realisation.[3]

Definition 4 *Let* $Z_n = \langle z_1, z_2, \ldots, z_n \rangle$ *be a set of* n *propositional variables. An* $\{\wedge, \vee, \neg\}$*-formula (or simply formula),* $\varphi(Z_n)$*, is any expression formed by a finite number of applications of the following rules:*

1. $\forall\, z \in Z_n$*,* z *and* $\neg z$ *are formulae.*
2. *If* ψ_1 *and* ψ_2 *are formulae then* $(\psi_1 \vee \psi_2)$ *and* $(\psi_1 \wedge \psi_2)$ *are both formulae.*

The size *of* $\varphi(Z_n)$*, denoted* $|\varphi(Z_n)|$*, is one plus the total number of applications of (2) in constructing it, i.e.* $|z| = |\neg z| = 1$ *and* $|\psi_1\ \theta\ \psi_2| = 1 + |\psi_1| + |\psi_2|$ *(for* $\theta \in \{\wedge, \vee\}$*).*

It should be noted that we do not *explicitly* allow the constant values $\{\top, \bot\}$ as formulae.

Propositional formulae (over $\{\wedge, \vee, \neg\}$) may be viewed as a restricted class of *Boolean networks* (also known as *straight-line programs*).

Definition 5 *A Boolean network* $C(V, E)$ *over the basis* $\{\wedge, \vee, \neg\}$ *and inputs* $\langle z_1, \ldots, z_n \rangle$ *is defined by a directed acyclic graph in which no vertex has in-degree exceeding* 2*. The vertices of* C *are either* gates *whose in-degree is* 2*; or* inputs *which have in-degree* 0*. The network* C *has exactly* $2n$ *input vertices each mapped to a unique member of* $\langle z_1, \neg z_1, z_2, \neg z_2 \ldots, z_n, \neg z_n \rangle$ *while gates are mapped to one of the operations* $\{\wedge, \vee\}$ *(when the in-degree is* 2*). Exactly one vertex (called the output) has out-degree* 0 *(note that we assume every input vertex has out-degree at least* 1*, otherwise the corresponding literal cannot influence the network computation). A* topological sort *of the vertices in a Boolean network* C *is a mapping* $\tau : V \rightarrow \{1, 2, \ldots, |V|\}$ *with the following properties:*

a. The input vertex associated with z_i *has* $\tau(z_i) = i$*.*

b. The input vertex associated with $\neg z_i$ *has* $\tau(z_i) = n + i$*.*

c. If w *is a* $\wedge$*-gate or* $\vee$*-gate vertex with* $\langle u, w \rangle \in E$ *and* $\langle v, w \rangle \in E$ *then* $\tau(w) > \max\{\tau(u), \tau(v)\}$*.*

[3] By which we mean that within the field of Boolean function complexity theory, see e.g. [Dunne, 1988], this model has a similar status to that of Turing machines in standard computational complexity theory.

It is well-known (e.g. using depth-first search) that a topological sort labelling always exists and can be constructed in $O(|V| + |E|)$ steps. Given $C(V, E)$ let $\langle z_1, \ldots, z_n, \neg z_1, \ldots, \neg z_n \rangle \subset V$ denote its inputs and $\langle g_1, g_2, \ldots, g_m \rangle$ its gates where C may be topologically sorted so that $\tau(g_i) = 2n + i$. Notice that g_m is the unique output. A Boolean network represents the propositional function $f : \langle \top, \bot \rangle^n \to \langle \top, \bot \rangle$ at g_m via: $res(g_m) = f(z_1, \ldots, z_n)$ where for $u \in V$,

$$res(u) = \begin{cases} z_i & \text{if } u \text{ is the input } z_i, \\ & \text{i.e. } \tau(u) = i \\ \neg z_i & \text{if } u \text{ is the input } \neg z_i, \\ & \text{i.e. } \tau(u) = n + i \\ res(v)\ \theta\ res(w) & \text{if } \{\langle v, u \rangle,\ \langle w, u \rangle\} \subseteq E \text{ and} \\ & u \text{ is an } \theta\text{-gate } (\theta \in \{\wedge, \vee\}) \end{cases}$$

The size *of a Boolean network, $C(V, E)$ with inputs $\langle z_1, \ldots, z_n \rangle$ is the number of* gate *vertices in V, i.e. $|V| - 2n$.*

One view of propositional *formulae* is as Boolean networks in which the out-degree of any *gate* vertex is at most 1.

4 Efficient computation of ADF models via AFs

We now present the main result of this paper: that for any ADF, $\langle \mathcal{S}, \mathcal{L}, \mathcal{C} \rangle$, there is an AF, $\langle \mathcal{X}, \mathcal{A} \rangle$, simulating it, in the sense of Defn. 3 *and* for which $|\mathcal{X}|$ is linear in the total size of the representations used for $\mathcal{C}$. Noting the view of formulae as a restricted type of Boolean network, without loss of generality, it is assumed that each acceptance condition $C_s \in \mathcal{C}$ is described by a network, $C_s(V_s, E_s)$.

The first step, presented in Lemma 1 shows that we can transform an arbitrary Boolean network into a corresponding AF. We then use this construction as the basis of the ADF simulation described in Thm. 1.

Lemma 1 *Let $C(V, E)$ be a Boolean network with inputs $\langle z_1, \ldots, z_n, \neg z_1, \ldots, \neg z_n \rangle$, output gate g_m (so that $|V| = 2n + m$) and using (gate) operations from $\{\wedge, \vee\}$. Suppose C realises the propositional function $f(z_1, \ldots, z_n)$, i.e. $res(g_m) = f(z_1, \ldots, z_m)$. There is an AF, $\langle \mathcal{X}, \mathcal{A} \rangle$ such that: 1) $\mathcal{X}$ contains arguments labelled $\{z_1, \ldots, z_n, \neg z_1 \ldots, \neg z_n, g_1, \ldots, g_m\}$; 2) For every assignment $\alpha \in \langle \top, \bot \rangle^n$ ($\alpha = \langle \alpha_1, \ldots, \alpha_n \rangle$) and vertex v with $res(v)(\alpha) = \top$, there is a subset Y of $\mathcal{X} \setminus \{z_1, \ldots, z_n, \neg z_1, \ldots, \neg z_n\}$ such that the set S^v_α given by $Y \cup \{v\} \cup \{\, z_i : \alpha_i = \top\} \cup \{\neg z_i : \alpha_i = \bot\}$ is an admissible set of $\langle \mathcal{X}, \mathcal{A} \rangle$. Furthermore if $res(g_m)(\alpha) = f(\alpha) = \top$ then $S^{g_m}_\alpha$ is a* stable extension *of $\langle \mathcal{X}, \mathcal{A} \rangle$. 3) For any stable extension $S \subseteq \mathcal{X}$ the assignment $\alpha_i = \top$ if $z_i \in S$; $\alpha_i = \bot$ if $\neg z_i \in S$ is well-defined (that is, S must contain* exactly *one of $\{z_i, \neg z_i\}$ for each $1 \leq i \leq n$) and is such that $g_m \in S$ if and only if $f(\alpha) = \top$. 4) $|\mathcal{X}| = 2n + O(m)$.*

Proof: We form a sequence of AFs, $\{\langle \mathcal{X}_i, \mathcal{A}_i \rangle\}_{0 \leq i \leq m}$ with:

$$\{z_1, \ldots, z_n, \neg z_1 \ldots, \neg z_n, g_1, \ldots, g_i\} \subseteq \mathcal{X}_i \subseteq \mathcal{X}_{i+1}$$
$$\mathcal{A}_i \subseteq \mathcal{A}_{i+1}$$

and $\langle \mathcal{X}_i, \mathcal{A}_i \rangle$ satisfies (1)–(4) wrt the sub-network of C induced by

$$\{z_1, \ldots, z_n, \neg z_1, \ldots, \neg z_n, g_1, \ldots, g_i\}$$

For the inductive base we have $i = 0$ and there are only input vertices of C to consider.

$$\begin{array}{lcl} \mathcal{X}_0 & := & \{z_1, \ldots, z_n, \neg z_1 \ldots, \neg z_n\} \\ \mathcal{A}_0 & := & \{\langle z_i, \neg z_i \rangle,\ \langle \neg z_i, z_i \rangle\ :\ 1 \leq i \leq n\} \end{array}$$

It is easily seen that $\langle \mathcal{X}_0, \mathcal{A}_0 \rangle$ already satisfies (1)–(4) (condition (3) being redundant since no gate arguments are present) as $res(z_i) = \top$ if and only if $z_i = \top$ (similarly $res(\neg z_i) = \top$ if and only if $z_i = \bot$).

For the inductive step, suppose we have constructed $\langle \mathcal{X}_{i-1}, \mathcal{A}_{i-1} \rangle$ with the desired properties. Consider, therefore g_i ($i \geq 1$) and the inductively defined AF, $\langle \mathcal{X}_{i-1}, \mathcal{A}_{i-1} \rangle$. Suppose that $\langle u_j, g_i \rangle \in E$ and $\langle u_k, g_i \rangle \in E$. Since we are observing topological order, by the inductive hypothesis it follows that $\{u_j, u_k\} \subseteq \mathcal{X}_{i-1}$.

Case 1: g_i is an $\wedge$-gate in $C(V, E)$.

$$\begin{array}{lcl} \mathcal{X}_i & := & \mathcal{X}_{i-1} \cup \{g_i, h^1_i, h^2_i\} \\ \mathcal{A}_i & := & \mathcal{A}_{i-1} \cup \{\langle u_j, h^1_i \rangle,\ \langle u_k, h^2_i \rangle,\ \langle h^1_i, g_i \rangle,\ \langle h^2_i, g_i \rangle\} \end{array}$$

In this case $res(g_i)(\alpha) = \top$ if and only if $res(u_j)(\alpha) = \top$ and $res(u_k)(\alpha) = \top$. It is immediate that (1) is satisfied; furthermore (2) holds: given α with $res(g_i)(\alpha) = \top$ let Z_α be given by $\{\, z_i\ :\ \alpha_i = \top\} \cup \{\, \neg z_i\ :\ \alpha_i = \bot\}$, Y_j be the subset defined from the fact $res(u_j)(\alpha) = \top$ and Y_k that from $res(u_k)(\alpha) = \top$. Then it can be shown that $Y_j \cup Y_k \cup \{g_i\} \cup Z_\alpha$ is is conflict-free and, thus, a stable extension of $\langle \mathcal{X}_i, \mathcal{A}_i \rangle$: the arguments $\{h^1_i, h^2_i\}$ being attacked by $\{u_j, u_k\}$. Condition (3) holds by considering any stable extension S of $\langle \mathcal{X}_i, \mathcal{A}_i \rangle$ for which $g_i \notin S$. In this case $S \cap \{h^1_i, h^2_i\} \neq \emptyset$ thus at least one of $u_j \notin S$ or $u_k \notin S$ holds. Let α be the assignment to $\langle z_1, \ldots, z_n \rangle$ arising from S: that $res(g_i)(\alpha) = \bot$ follows by induction using the subset of S defining a stable extension in $\langle \mathcal{X}_j, \mathcal{A}_j \rangle$ ($u_j \notin S$) or $\langle \mathcal{X}_k, \mathcal{A}_k \rangle$ (when $u_k \notin S$).

Finally $|\mathcal{X}_i| = |\mathcal{X}_{i-1}| + 3$ establishing that (4) holds.

Case 2: g_i is an $\vee$-gate in $C(V, E)$.

$$\begin{array}{lcl} \mathcal{X}_i & := & \mathcal{X}_{i-1} \cup \{g_i, p_i\} \\ \mathcal{A}_i & := & \mathcal{A}_{i-1} \cup \{\langle u_j, p_i \rangle,\ \langle u_k, p_i \rangle,\ \langle p_i, g_i \rangle\} \end{array}$$

From $res(g_i)(\alpha) = \top$ if and only if at least one of $res(u_j)(\alpha) = \top$ or $res(u_k)(\alpha) = \top$, a similar argument to the $\wedge$-gate case establishes this case. □

Theorem 1 *Let $\langle \mathcal{S}, \mathcal{L}, \mathcal{C} \rangle$ be an ADF and assume that each acceptance condition $C_i \in \mathcal{C}$ is described using a Boolean network, (V_i, E_i) with output gate s_i such that $res(s_i)(par(s_i)) = C_i$. There is an AF, $\langle \mathcal{X}, \mathcal{A} \rangle$ that: simulates $\langle \mathcal{S}, \mathcal{L}, \mathcal{C} \rangle$; has $|\mathcal{X}|$ linear in the total size of networks encoding $\mathcal{C}$; and is constructible in polynomial time.*

Proof: For any $s \in \mathcal{S}$ for which $par(s) = \emptyset$, we have $C_s(par(s)) \in \{\top, \bot\}$ and the AFs in Fig 1 are used (note that these are not dealt with in Lemma 1).

For the remaining cases, recalling that each acceptance condition, C_s, describes a propositional function of the variables $par(s)$, first construct, for each $s \in \mathcal{S}$ an AF, $\langle \mathcal{X}_s, \mathcal{A}_s \rangle$ with the properties described in Lemma 1. In these AFs, let s be the argument (of $\mathcal{X}_s$) corresponding to the output gate of (V_s, E_s) and $\langle r_1, \ldots, r_t, \neg r_1, \ldots, \neg r_t \rangle$ be the arguments defining the inputs of (V_s, E_s) (so that $par(s) =$

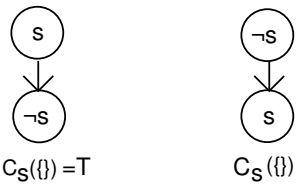


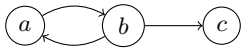
Figure 1: AFs simulating $C_s(\emptyset) \in \langle \top, \bot \rangle$

$\{r_1, \ldots, r_t\}$ in $\langle \mathcal{S}, \mathcal{L}, \mathcal{C} \rangle$). It is convenient (although not essential) to add for each (non-constant) case, an additional argument to $\langle \mathcal{X}_s, \mathcal{A}_s \rangle$ which is denoted $\neg s$ and the attacks $\{\langle \neg s, s \rangle, \langle s, \neg s \rangle\}$.

To complete the construction, for each $\langle \mathcal{X}_s, \mathcal{A}_s \rangle$ we identify the input arguments, i.e. $\langle r_1, \ldots, r_t, \neg r_1, \ldots, \neg r_t \rangle$ with the respective output arguments, i.e from $\langle \mathcal{X}_{r_j}, \mathcal{A}_{r_j} \rangle$. The only exception is when C_r is a constant function in which case the mutual attack $\{\langle r, \neg r \rangle, \langle \neg r, r \rangle\}$ in $\langle \mathcal{X}_s, \mathcal{A}_s \rangle$ is removed and the appropriate structure from Fig 1 used instead. Notice that one property of the construction is that an argument p corresponding to the output of (V_p, E_p) – the network encoding C_p – is never in conflict with any argument q corresponding to the output of (V_q, E_q) even when $\langle p, q \rangle \in \mathcal{L}$ so that p is one of the input node of (V_q, E_q). Suppose that $M \subseteq \mathcal{S}$ is a model of $\langle \mathcal{S}, \mathcal{L}, \mathcal{C} \rangle$. By definition, M is conflict-free and the corresponding subset $S_M = \{ s \in \mathcal{X} : s \in M \}$ is also conflict-free in $\langle \mathcal{X}, \mathcal{A} \rangle$ from our earlier observations. It remains to construct a stable extension, T_M in $\langle \mathcal{X}, \mathcal{A} \rangle$ with $S_M \subseteq T_M$. From the fact that M is a model in $\langle \mathcal{S}, \mathcal{L}, \mathcal{C} \rangle$ we have $s \in M$ if and only if $C_s[M \cap par(s)] = \top$. It follows that by considering (V_s, E_s) the Boolean network encoding of C_s and the inputs from $M \cap par(s)$, that the assignment $\alpha_{M \cap par(s)}$ to $par(s)$ is such that $res(g_s)(\alpha_{M \cap par(s)}) = \top$. Thus, via Lemma 1, for every $s \in M$ we find a stable extension Y_s in $\langle \mathcal{X}_s, \mathcal{A}_s \rangle$. It follows, via a similar argument to that used previously, that $M_S \cup \bigcup_{s \in M} Y_s$ is a stable extension of $\langle \mathcal{X}, \mathcal{A} \rangle$ completing the proof of (b). For space reasons we omit the proof that if T is stable extensions of $\langle \mathcal{X}, \mathcal{A} \rangle$ then $T \cap \mathcal{S}$ is a model. □

We conclude this section with a small illustrating example.

Example 1 *Consider the* ADF D *given as* $\langle \{a, b, c\}, \{\langle a, b \rangle, \langle b, a \rangle, \langle b, c \rangle\}, \{C_a, C_b, C_c\} \rangle$*, i.e.*



In this $C_a(b) = b$*,* $C_b(a) = a$ *and* $C_c(b) = \neg b$*. The* AF*,* H^D *adds to* $\{a, b, c\}$ *two auxiliary arguments –* z_a *and* z_b *– and has attack set* $\{\langle z_a, a \rangle, \langle a, z_b \rangle, \langle z_b, b \rangle, \langle b, z_a \rangle, \langle b, c \rangle\}$*.* H^D *has two stable extensions –* $\{a, b\}$ *and* $\{z_a, z_b, c\}$ *– the former corresponding to the model* $\{a, b\}$ *and the latter to the model* $\{c\}$*.*

5 Grounded and stable ADF models

As we have seen, models of an ADF D can be computed efficiently by translating D into an argumentation framework H^D. The stable models of the latter exactly characterize the models of D. Now, although models certainly are a fundamental notion in the ADF approach, they are not the most significant semantic notions. As for AFs, well-founded, stable and preferred semantics appear far more interesting. For this reason we extend our analysis, focusing on well-founded and stable models.[4]

5.1 Well-founded models

We recall that the *well-founded* model of an ADF, $D = \langle \mathcal{S}, \mathcal{L}, \mathcal{C} \rangle$ is defined via the operator Γ_D with $\Gamma_D(A, R) = (\ acc(A, R),\ reb(A, R)\ )$ where $acc(A, R)$ is $\{r \in \mathcal{S} : A \subseteq S' \subseteq \mathcal{S} \setminus R \Rightarrow C_r[S' \cap par(r)] = \top\}$ and $reb(A, R)$ is $\{r \in \mathcal{S} : A \subseteq S' \subseteq \mathcal{S} \setminus R \Rightarrow C[S' \cap par(r)] = \bot]\}$. As shown in [Brewka and Woltran, 2010], it has a least fixed point obtained by iterating $\Gamma_D(\emptyset, \emptyset)$. The well-founded model of D is defined as the set $E \subseteq \mathcal{S}$ for which there is some $E' \subseteq \mathcal{S}$ having (E, E') the least fixed point of Γ_D.

It turns out that the AF simulating an ADF captures the well-founded model via its grounded extension in the same way as it captures ADF models via its stable models.

Theorem 2 *Let* $D = \langle S, L, C \rangle$ *be an* ADF *with well-founded model* W*. Let* H^D *the* AF *resulting from translating* D*,* G *its grounded extension. Then we have* $G \cap S = W$*.*

5.2 Stable models: loop formulas

We first recall the definition of stable models from [Brewka and Woltran, 2010]. They are defined for bipolar ADFs (BADFs) only where each link is supporting or attacking: formally a link, (r, s) is supporting (attacking) iff there is no $R \subseteq par(s)$ for which $C_s(R) = \top$ (resp. $\bot$) but $C_s(R \cup \{r\}) = \bot$ (resp. $\top$).

Let $D = \langle S, L, C \rangle$ be a BADF. A model M of D is called *stable* if M is the least model[5] of the reduced ADF D^M obtained from D by

1. eliminating all nodes not contained in M together with all links in which any of these nodes appear,
2. eliminating all attacking links,
3. replacing in each acceptance condition C_s of a node s in D^M each occurrence of a statement $t \notin M$ with $\bot$.

This definition, inspired by the definition of the same notion for logic programs [Gelfond and Lifschitz, 1991], excludes models containing self-supporting cycles, i.e., each node in a stable model is required to have independent support.

Rather than modifying the simulating AF to exclude unwanted, non-stable models, we stay at the ADF level and proceed in 2 steps as follows: we first show how an arbitrary BADF D can be translated into an ADF D^* such that the stable models of D coincide with the models of D^*. We then simulate D^* using the translation described above. We thus obtain an AF H^{D^*} whose stable models are in one-to-one correspondence with the stable models of D.

For the construction of D^* we will adapt a technique based on so-called loop formulas, originally proposed by Lin and Zhao [2004] for logic programming. Lin and Zhao showed that by adding adequate formulas to the Clark completion of

[4] Preferred models depend on the definition of stable models and thus may be considered somewhat less fundamental.

[5] There is a unique such model [Brewka and Woltran, 2010].

a logic program one can obtain a propositional theory whose models are exactly the stable models of the original program.

The definition of the loop formulas required for BADFs is simplified because (1) supporting cycles can directly be "read off" the BADF graph, and (2) an explicit acceptance condition for each node, rather than a set of rules, is given.

Let D be a BADF. We call a non-empty set $L = \{A_1, \ldots, A_n\}$ of nodes in D a support loop whenever $A_i, A_j \in L$ implies that A_j is reachable from A_i through a (nonempty) sequence of supporting links in D.

Let $L = \{A_1, \ldots, A_n\}$ be a support loop and $C_1, \ldots, C_n$ the acceptance conditions of $A_1, \ldots, A_n$, respectively. The loop formula of L is the implication

$$A_1 \wedge \ldots \wedge A_n \rightarrow C_1^{\neg L} \vee C_2^{\neg L} \vee \ldots \vee C_n^{\neg L}.$$

Here, $C_i^{\neg L}$ stands for the formula which is obtained by replacing in the acceptance condition C_i each occurrence of any element in L by $\bot$. Intuitively, the implication says that when the nodes in L are $\top$, then there must be independent support for at least one of the nodes A_i in L. Independent support means: A_i's acceptance condition C_i is satisfied even if the predecessors of A_i in the support loop are not $\top$.

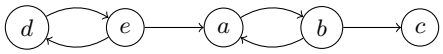
For the construction of D^* we need to compute the loop formulas of all support loops in D. Let $LF_1, \ldots, LF_k$ be the (complete) collection of the loop formulas.

We must guarantee that D^* has no model in which any of the loop formulas is violated. This can be achieved by introducing a new, self-attacking node, say F. The self-attack must become "active" whenever one of the loop formulas is violated. This can be achieved by choosing for F the following acceptance condition:

$$C_F : (\neg LF_1 \vee \ldots \vee \neg LF_k) \wedge \neg F.$$

This is similar to the use of constraints in logic programming. As long as no loop formula is violated, F cannot be $\top$ and does not do any harm. However, if there is a violation, then the respective set of nodes cannot be a model, as F is missing although C_F is satisfied; yet adding F also does not lead to a model, as $\neg F$ and thus C_F is then not satisfied.

Example 2 *Consider the* BADF *given in Example 1 above. Recall links (a, b) and (b, a) are supporting (that is, $C_a = b$, $C_b = a$) and (b, c) is attacking ($C_c = \neg b$). The single support loop is $\{a, b\}$. Its loop formula is $a \wedge b \rightarrow \bot$ which is equivalent to $\neg(a \wedge b)$. To construct D^* we introduce a new node F with $C_F = a \wedge b \wedge \neg F$. It is easy to verify that the single model of D^* is exactly the single stable model of D. Now consider the extended* BADF *D':*



Links (e, d) and (d, e) are attacking ($C_e = \neg d$, $C_d = \neg e$). (e, a) is an additional supporting link for a and C_a becomes $e \vee b$. D' has three models, namely $M_1 = \{e, a, b\}$, $M_2 = \{d, c\}$ and $M_3 = \{d, a, b\}$. M_1 as well as M_2 are stable, M_3 is not.

As before, the single support loop is $\{a, b\}$. Its loop formula (after simplifications) now is $a \wedge b \rightarrow (e \vee \bot)$. To construct D'^* we add a node F with $C_F = a \wedge b \wedge \neg e \wedge \neg F$. M_1 and M_2 are models of D'^* (the acceptance condition of F is not satisfied). As intended, M_3 is not a model as the self-attack of F is enforced whenever a, b are $\top$ and e is $\bot$.

Theorem 3 *Let D be a* BADF*, D^* the* ADF *constructed as described above. The stable models of D are exactly the models of D^*.*

Although D^* has only a single additional node F, the size of C_F may obviously be exponential as there may be an exponential number of loops. The (indirect) translation to A^{D^*} thus may lead to an exponential blowup. We thus present next an alternative translation which avoids such a blowup.

5.3 Stable models: a polynomial translation

The idea here is to cope with self-supporting loops by observing that the existence of a certain order for checking the satisfaction of the acceptance excludes such self-supports. We can in fact use ADFs themselves for guessing such orders, and in case an order exists, the models of a suitably extended ADF will correspond to the stable models of the original ADF. Thus compared to the previous approach which followed the ideas of loop formulas, we generalize here another concept from logic programs, viz. level mappings as, for instance, used in [Ben-Eliyahu and Dechter, 1994].

For this purpose we need a certain ADF for guessing orders between statements S. To this end, we define the ADF

$$D_S^{<} = (S(1) \cup \cdots \cup S(n) \cup S^{<}, L, C)$$

where n is the cardinality of S, $S(i) = \{s^i \ : \ s \in S\}$ carries a copy s^i for each $s \in S$, and $S^{<} = \{f\} \cup \{(t < s) \ : \ s, t \in S, s \neq t\}$ provides statements $(t < s)$ which are used to indicate an order between elements from S and f will be a "constraint" statement which ensures the order. We next describe the parts of L and C. First we have links between all s^i for a given i, i.e. $\{(s^i, t^i) \ : \ s, t \in S, s \neq t\} \subseteq L$ for each i. All these links are simply attacking, thus at most one s^i is selected for each level i in a model. In order to ensure that exactly one such s^i is selected and that for each i different statements are selected, we use the constraint f, i.e. we have links in L from each s^i to f and the acceptance condition for f is given as

$$\Big(\big(\bigvee_{i=1}^{n} \bigwedge_{s \in S} \neg s^i \big) \vee \big(\bigvee_{i=1}^{n-1} \bigvee_{j=i+1}^{n} \bigvee_{s \in S} (s^i \wedge s^j) \big) \Big) \wedge \neg f.$$

Finally, for each statement $(t < s)$ we link all s^i's and t^i's to $(t < s)$ and define as acceptance condition for $(t < s)$:

$$\bigwedge_{i=1}^{n} (s^i \rightarrow \bigvee_{j=1}^{i-1} t^j).$$

Thus, $(t < s)$ has to be in a model iff t^j and s^i with $j < i$ are in the model. We note that for given S, $D_S^{<}$ can be constructed from S in polynomial time.

Lemma 2 *Let S be a set of statements and $D_S^{<}$ as given above. Then, (i) for each order $<$ on S, there exists a model M of $D_S^{<}$ such that $(t < s) \in M \cap S^{<}$ iff $t < s$ holds; (ii) for each model M of $D_S^{<}$ the statements in $M \cap S^{<}$ describe a valid total order on S.*

We are now ready to define a translation which maps an ADF $D = \langle S, L, C\rangle$ to an ADF $D^\dagger$ such that the stable models of D are in correspondence with the models of $D^\dagger$. The construction of $D^\dagger$ is as follows:

1. it contains the original ADF D;
2. it contains a certain copy D' of D which is linked to
3. the order-guessing ADF $D_S^<$; and
4. has a further constraint node g linked from D and D'.

The copy $D' = \langle S', L', C'\rangle$ is the same as D but for each supporting link (t, s) in D there is now also a link from $(t < s)$ to s'. Accordingly the acceptance conditions $C'_{s'}$ are obtained from C_s by replacing each occurrence of t (where (t, s) is supporting in D) by $(t' \wedge (t < s))$, and all remaining occurrences of statements u (i.e. (u, s) is attacking in D) by u'. Finally, the acceptance condition for the additional node g is:

$$\Big(\bigvee_{s\in S}\big((s \wedge \neg s') \vee (\neg s \wedge s')\big)\Big) \wedge \neg g.$$

Again, $D^\dagger$ can be constructed from D in polynomial time.

Theorem 4 *For any* BADF $D = \langle \mathcal{S}, \mathcal{L}, \mathcal{C}\rangle$*, the stable models of* D *are given by* $\{M \cap S \ : \ M$ *is a model of* $D^\dagger\}$.

We thus can compute stable models of D by constructing the AF $H^{D^\dagger}$. Each stable model of the latter will contain a stable model of D, and each stable model of D will be contained in a model of $H^{D^\dagger}$. Contrary to the loop formula approach this construction is polynomial. This comes at a cost, though: the models of $H^{D^\dagger}$ contain numerous nodes whose intuitive meaning is entirely technical and not interpretable in terms of the original statements in D.

6 Discussion and Conclusions

Our main aim in this paper has been to consider the relationship between the ADF model of abstract argumentation from [Brewka and Woltran, 2010] and the widely studied approach of argumentation frameworks from [Dung, 1995]. An important motivation for ADFs (as indeed had earlier been the case for bipolar AFs) concerns the restrictive view of argument interaction within AFs: in AFs arguments are related only in terms of one *attacking* another. Bipolar frameworks recognize that arguments may interact in other ways (e.g. through notions of support); ADFs continue and enrich the sphere of *explicitly* recognized interaction between arguments through their use of specified propositional acceptance conditions.

Our simulation shows that computation of ADF *models* is equivalent to computing the stable extensions in a related AF. For well-founded and stable models of ADFs we obtain a similar picture. The well-founded model is captured by the grounded extension of the related AF. For stable models the situation is somewhat more involved: we presented two ADF-to-ADF translations which provide a relation between stable models of an ADF and models of its translation. The translation based on loop formulas was exact yet exponential, the non-exact translation introduces numerous nodes whose meaning is entirely technical. In total the results may be interpreted as demonstrating that *arbitrarily complex* interactions describing the acceptability of an argument x in terms of acceptability of its parent arguments, $par(x)$, *can always* be expressed in terms of *attacks* between arguments in an AF $\langle \mathcal{Y} \cup \{x\} \cup par(x), \mathcal{A}\rangle$. Of course while in principle it is sufficient to consider only "attack" as the basis for argument interaction, this will often render direct modeling of specific argumentation scenarios rather opaque: this is most apparent for stable models where users must specify arguments whose purpose is entirely technical.

In this light, noting contributions such as that of [Brewka and Gordon, 2010] in reconstructing the less abstracted and more natural expressive formalism defined by the Carneades framework of [Gordon *et al.*, 2007], we can view our results in a much more positive light: since ADFs can be "compiled" into "equivalent" AFs with polynomial overhead, ADFs can be used as a convenient representation device but examined and analysed at the basic AF level. One issue, and the subject of current empirical study, is the extent to which the resulting AF structures have reasonable algorithmic properties.

References

[Amgoud *et al.*, 2008] L. Amgoud, C. Cayrol, M.-C. Lagasquie-Schiex, and P. Livet. On bipolarity in argumentation frameworks. *Int. J. Intell. Syst.*, 23(10):1062–1093, 2008.

[Ben-Eliyahu and Dechter, 1994] R. Ben Eliyahu and R. Dechter. Propositional semantics for disjunctive logic programs *Ann. Math. Artif. Intell.* 12(1-2): 53–87, 1994.

[Brewka and Gordon, 2010] G. Brewka and T. F. Gordon. Carneades and abstract dialectical frameworks: A reconstruction. In *Proc. 3rd COMMA*, volume 216 of *FAIA*, pages 3–12. IOS Press, 2010.

[Brewka and Woltran, 2010] G. Brewka and S. Woltran. Abstract dialectical frameworks. In *Proc. KR 2010*, pages 102–111, 2010.

[Cayrol and Lagasquie-Schiex, 2005] C. Cayrol and M.-C. Lagasquie-Schiex. On the acceptability of arguments in bipolar argumentation frameworks. In *Proc. 8th ECSQARU*, volume 3571 of *LNCS*, pages 378–389. 2005.

[Dung, 1995] P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming, and N-person games. *Artif. Intell.*, 77:321–357, 1995.

[Dunne, 1988] P. E. Dunne. *The Complexity of Boolean Networks*. Academic Press, 1988.

[Gabbay, 2009] D. M. Gabbay Fibring argumentation frames. *Studia Logica*, 93:231–295, 2009

[Gelfond and Lifschitz, 1991] M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Comput.*, 9:365–386, 1991.

[Gordon *et al.*, 2007] T. F. Gordon, H. Prakken, and D. Walton. The Carneades model of argument and burden of proof. *Artif. Intell.*, 171:875–896, 2007.

[Lin and Zhao, 2004] F. Lin and Y. Zhao. ASSAT: computing answer sets of a logic program by sat solvers. *Artif. Intell.*, 157(1-2):115–137, 2004.