On the Relative Expressiveness of Argumentation Frameworks, Normal Logic Programs and Abstract Dialectical Frameworks

Hannes Strass
Computer Science Institute
Leipzig University, Germany

Abstract

We analyse the expressiveness of the two-valued semantics of abstract argumentation frameworks, normal logic programs and abstract dialectical frameworks. By expressiveness we mean the ability to encode a desired set of two-valued interpretations over a given propositional signature using only atoms from that signature. While the computational complexity of the two-valued model existence problem for all these languages is (almost) the same, we show that the languages form a neat hierarchy with respect to their expressiveness.

Introduction

More often than not, different knowledge representation languages have conceptually similar and partially overlapping intended application areas. What are we to do if faced with an application and a choice of several possible knowledge representation languages which could be used for the application? One of the first axes along which to compare different formalisms that comes to mind is computational complexity: if a language is computationally too expensive when considering the problem sizes typically encountered in practice, then this is a clear criterion for exclusion.

But what if the available language candidates have the same computational complexity? If their expressiveness in the computational-complexity sense of "What kinds of *problems* can the formalism solve?" is the same, we need a more fine-grained notion of expressiveness. In this paper, we use such an alternative notion and perform an exemplary study of the relative expressiveness of several different knowledge representation languages: argumentation frameworks (AFs) (Dung, 1995), normal logic programs (LPs), abstract dialectical frameworks (ADFs) (Brewka and Woltran, 2010) and propositional logic.

This choice of languages is largely motivated by the similar intended application domains of argumentation frameworks and abstract dialectical frameworks and the close relation of the latter to normal logic programs. We add propositional logic to have a well-known reference point. Furthermore, the computational complexity of their respective model existence problems is the same (with one exception):

- for AFs, deciding stable extension existence is NP-complete (Dimopoulos, Nebel, and Toni, 2002);
- for LPs, deciding the existence of supported/stable models is NP-complete (Bidoit and Froidevaux, 1991; Marek and Truszczyński, 1991);
- for ADFs, deciding the existence of models is NP-complete (Brewka et al., 2013), deciding the existence of stable models is Σ_2^P-complete for general ADFs (Brewka et al., 2013) and NP-complete for the subclass of bipolar ADFs (Strass and Wallner, 2014);
- the satisfiability problem of propositional logic is NP-complete.

In view of these almost identical complexities, we use an alternative measure of the expressiveness of a knowledge representation language L: "Given a set of two-valued interpretations, is there a knowledge base in L that has this exact model set?" This notion lends itself straightforwardly to compare different formalisms (Gogic et al., 1995):

> Formalism L_2 is at least as expressive as formalism L_1 if and only if every knowledge base in L_1 has an equivalent knowledge base in L_2.

So here expressiveness is understood in terms of *realisability*, "What kinds of model sets can the formalism express?"

It is easy to see that propositional logic can express any set of two-valued interpretations. The same is easy (but less easy) to see for logic programs under supported model semantics. For logic programs under stable model semantics, it is clear that not all model sets can be expressed, since two different stable models are always incomparable with respect to the subset relation. In this paper, we study such expressiveness properties for all the mentioned formalisms under different semantics. It will turn out that the languages form a more or less strict expressiveness hierarchy, with AFs at the bottom, ADFs and LPs under stable semantics higher up and ADFs and LPs under supported model semantics at the top together with propositional logic.

To show that a language L_2 is at least as expressive as a language L_1 we will mainly use two different techniques. In the best case, we can use a syntactic compact and faithful translation from knowledge bases of L_1 to those of L_2. *Compact* means that the translation does not change the vocabulary, that is, does not introduce new atoms. *Faithful* means that the translation exactly preserves the models of the knowledge base for respective semantics of the two languages. In the second best case, we assume given the

knowledge base of L_1 in the form of a set X of desired models and construct a semantic *realisation* of X in L_2, that is, a knowledge base in L_2 whose model set corresponds exactly to X. To show that language L_2 is *strictly more expressive* than L_1, we additionally have to present a knowledge base K from L_2 of which we prove that L_1 cannot express the model set of K.

For all methods, we can make use of several recent works on the formalisms we study here. First of all, we [2013] studied the syntactic intertranslatability of ADFs and LPs, but did not look at expressiveness or realisability. The latter was recently studied for argumentation frameworks by Dunne et al. (2014). They allow to extend the vocabulary in order to realise a given model set, as long as the new vocabulary elements are evaluated to false in all models. For several semantics of AFs, Dunne et al. found necessary (and sufficient) conditions for realisability. While their sufficient conditions are not applicable to our setting, they discovered a necessary condition for realisability with stable extension semantics that we will make use of in this paper. There has also been work on translating ADFs into AFs for the ADF model and AF stable extension semantics (Brewka, Dunne, and Woltran, 2011), however this translation introduces additional arguments and is therefore not compact.

The gain that is achieved by our results is not only that of increased clarity about fundamental properties of these knowledge representation languages – *What can these formalisms express, actually?* – but has several further applications. As Dunne et al. (2014) remarked, a major application is in constructing knowledge bases with the aim of encoding a certain model set. As a necessary prerequisite to this, it must be known that the intended model set is realisable in the first place. For example, in a recent approach to revising argumentation frameworks (Coste-Marquis et al., 2013), the authors avoid this problem by assuming to produce a *collection* of AFs whose model sets in union produce the desired model set. While the work of Dunne et al. (2014) showed that this is indeed necessary in the case of AFs and stable extension semantics (that is, there are model sets that a single AF just cannot express), our work shows that for ADFs under the model semantics, a single knowledge base (ADF) is always enough to realise any given model set.

Of course, the fact that the languages we study have the same computational complexity means that there in principle exist polynomial intertranslations for the respective decision problems. But such intertranslations may involve the introduction of new atoms. In theory, a polynomial blowup from n atoms to n^k atoms for some k is of no consequence. In practice, it has a profound impact: the number n of atoms directly influences the search space that any implementation potentially has to cover. There, an increase from 2^n to 2^{n^k} is no longer polynomial, but exponential, and accordingly makes itself felt. Being able to realise a model set compactly, without new atoms, therefore attests that a language L has a certain basic kind of efficiency property, in the sense that the L-realisation of a model set does not unnecessarily enlarge the search space of algorithms operating on it.

The paper proceeds as follows. We first define the notion of expressiveness formally and then introduce the languages we will study. After reviewing several intertranslatability results for these languages, we stepwise obtain the results that lead to the expressiveness hierarchy. We conclude with a discussion of avenues for future work.

Background

We assume given a finite set A of atoms (statements, arguments), the *vocabulary*. A knowledge representation language interpreted over A is then some set L; a (two-valued) semantics for L is a mapping $\sigma : L \to 2^{2^A}$ that assigns sets of two-valued models to the language elements. (So A is implicit in L.) Strictly speaking, a two-valued interpretation is a mapping from the set of atoms into the two truth values true and false, but for technical ease we represent two-valued interpretations by the sets containing the atoms that are true.

For a language L, we denote the range of the semantics σ by $\sigma(L)$. Intuitively, $\sigma(L)$ is the set of models that language L can express, with any knowledge base over vocabulary A whatsoever. For example, for $L = \text{PL}$ propositional logic and $\sigma = mod$ the usual model semantics, we have $\sigma(\text{PL}) = 2^{2^A}$ since obviously any set of models is realisable in propositional logic.[1] This leads us to compare different pairs of languages and semantics with respect to the semantics' range of models. Our concept of "language" concentrates on semantics and decidedly remains abstract.

Definition 1. Let A be a finite vocabulary, L_1, L_2 be languages that are interpreted over A and $\sigma_1 : L_1 \to 2^{2^A}$ and $\sigma_2 : L_2 \to 2^{2^A}$ be two-valued semantics. We define

$$L_1^{\sigma_1} \leq_e L_2^{\sigma_2} \quad \text{iff} \quad \sigma_1(L_1) \subseteq \sigma_2(L_2)$$

Intuitively, language L_2 under semantics σ_2 is at least as expressive as language L_1 under semantics σ_1, because all models that L_1 can express under σ_1 are also contained in those that L_2 can produce under σ_2. (If the semantics are clear from the context we will omit them; this holds in particular for argumentation frameworks and propositional logic, where we only look at a single semantics.) As usual,

- $L_1 <_e L_2$ iff $L_1 \leq_e L_2$ and $L_2 \not\leq_e L_1$;
- $L_1 \cong_e L_2$ iff $L_1 \leq_e L_2$ and $L_2 \leq_e L_1$.

The relation $\leq_e$ is reflexive and transitive by definition, but not necessarily antisymmetric. That is, there might different languages $L_1 \neq L_2$ that are equally expressive: $L_1 \cong_e L_2$.

We next introduce the particular knowledge representation languages we study in this paper. All will make use of a vocabulary A; the results of the paper are all considered parametric in such a given vocabulary.

Logic Programs

For a vocabulary A define $not\ A = \{not\ a \mid a \in A\}$ and the set of literals over A as $A^{\pm} = A \cup not\ A$. A *normal logic program rule* over A is then of the form $a \leftarrow B$ where $a \in A$ and $B \subseteq A^{\pm}$. The rule can be read as logical consequence, "a is true if all literals in B are true." The set B

[1] For a set $X \subseteq 2^A$ we can simply define $\varphi_X = \bigvee_{M \in X} \varphi_M$ with $\varphi_M = \bigwedge_{a \in M} a \wedge \bigwedge_{a \in A \setminus M} \neg a$ and clearly $mod(\varphi_X) = X$.

is called the *body* of the rule, we denote by $B^+ = B \cap A$ and $B^- = \{a \in A \mid not\ a \in B\}$ the *positive* and *negative body* atoms, respectively. A rule is *definite* if $B^- = \emptyset$. For singleton $B = \{b\}$ we denote the rule just by $a \leftarrow b$. A *logic program (LP)* P over A is a set of logic program rules over A, and it is definite if all rules in it are definite.

At first, logic programs were restricted to definite programs, whose semantics was defined through the proof-theoretic procedure of SLD resolution. The meaning of negation not was only defined operationally through negation as failure. Clark (1978) gave the first declarative semantics for normal logic programs via a translation to classical logic that will be recalled shortly. This leads to the supported model semantics for logic programs: A rule $a \leftarrow B \in P$ is *active* in a set $M \subseteq A$ iff $B^+ \subseteq M$ and $B^- \cap M = \emptyset$ imply $a \in M$. M is a *supported model* for P iff $M = \{a \in A \mid a \leftarrow B \in P \text{ is active in } M\}$. For a logic program P we denote the set of its supported models by $su(P)$. The intuition behind this semantics is that everything that is true in a model has some kind of support.

However, this support might be cyclic self-support. For instance, the logic program $\{a \leftarrow a\}$ has two supported models, $\emptyset$ and $\{a\}$, where the latter is undesired in many application domains. As an alternative, Gelfond and Lifschitz (1988) proposed the stable model semantics, a declarative semantics for negation as failure that does not allow self-support: $M \subseteq A$ is a *stable model* for P iff M is the $\subseteq$-least supported model of P^M, where the definite program P^M is obtained from P by (1) eliminating each rule whose body contains a literal $not\ a$ with $a \in M$, and (2) deleting all literals of the form $not\ a$ from the bodies of the remaining rules. We write $st(P)$ for the set of stable models of P. It follows from the definition of stable models that $st(P)$ is a $\subseteq$-antichain: for all $M_1 \neq M_2 \in st(P)$ we have $M_1 \not\subseteq M_2$.

Argumentation Frameworks

Dung (1995) introduced argumentation frameworks as pairs $F = (A, R)$ where A is a set and $R \subseteq A \times A$ a relation. The intended reading of an AF F is that the elements of A are arguments whose internal structure is abstracted away. The only information about the arguments is given by the relation R encoding a notion of attack: a pair $(a, b) \in R$ expresses that argument a attacks argument b in some sense.

The purpose of semantics for argumentation frameworks is to determine sets of arguments (called *extensions*) which are acceptable according to various standards. For a given extension $S \subseteq A$, the arguments in S are considered to be accepted, those that are attacked by some argument in S are considered to be rejected, and all others are neither, their status is undecided. We will only be interested in so-called *stable* extensions, sets S of arguments that do not attack each other and attack all arguments not in the set. For stable extensions, each argument is either accepted or rejected by definition, thus the semantics is two-valued. More formally, a set $S \subseteq A$ of arguments is *conflict-free* iff there are no $a, b \in S$ with $(a, b) \in R$. A set S is a *stable extension* for (A, R) iff it is conflict-free and for all $a \in A \setminus S$ there is a $b \in S$ with $(b, a) \in R$. For an AF F, we denote the set of its stable extensions by $st(F)$. Again, it follows from the definition of a stable extension that the set $st(F)$ is always a $\subseteq$-antichain.

Abstract Dialectical Frameworks

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

More formally, an *abstract dialectical framework* is a tuple $D = (A, L, C)$ where

- A is a set of statements,
- $L \subseteq A \times A$ is a set of links,
- $C = \{C_a\}_{a \in A}$ is a collection of total functions $C_a : 2^{par(a)} \to \{\mathbf{t}, \mathbf{f}\}$, one for each statement a. The function C_a is called *acceptance condition of* a.

It is often convenient to represent acceptance conditions by propositional formulas. In particular, we will do so for several results of this paper. There, each C_a is represented by a propositional formula φ_a over $par(a)$. Then, clearly, $C_a(R \cap par(a)) = \mathbf{t}$ iff R is a model for φ_a, $R \models \varphi_a$.

Brewka and Woltran (2010) introduced a useful subclass of ADFs: an ADF $D = (A, L, C)$ is *bipolar* iff all links in L are supporting or attacking (or both). A link $(b, a) \in L$ is *supporting in* D iff for all $R \subseteq par(a)$, we have that $C_a(R) = \mathbf{t}$ implies $C_a(R \cup \{b\}) = \mathbf{t}$. Symmetrically, a link $(b, a) \in L$ is *attacking in* D iff for all $R \subseteq par(a)$, we have that $C_a(R \cup \{b\}) = \mathbf{t}$ implies $C_a(R) = \mathbf{t}$. If a link (b, a) is both supporting and attacking then b has no influence on a, the link is redundant (but does not violate bipolarity). We will sometimes use this circumstance when searching for ADFs; there we simply assume that $L = A \times A$, then links that are actually not needed can be expressed by acceptance conditions that make them redundant.

There are numerous semantics for ADFs; we will only be interested in two of them, (supported) models and stable models. A set $M \subseteq A$ is a *model of* D iff for all $a \in A$ we find that $a \in M$ iff $C_a(M) = \mathbf{t}$. The definition of stable models is inspired by logic programming and slightly more complicated (Brewka et al., 2013). Define an operator by $\Gamma_D(Q, R) = (acc(Q, R), rej(Q, R))$ for $Q, R \subseteq A$, where

$$acc(Q, R) = \{a \in A \mid \text{for all } Q \subseteq Z \subseteq (A \setminus R), \text{ we have } C_a(Z) = \mathbf{t}\}$$
$$rej(Q, R) = \{a \in A \mid \text{for all } Q \subseteq Z \subseteq (A \setminus R), \text{ we have } C_a(Z) = \mathbf{f}\}$$

The intuition behind the operator is as follows: A pair (Q, R) represents a partial interpretation of the set of statements where those in Q are accepted (true), those in R are rejected (false), and those in $S \setminus (Q \cup R)$ are neither.

The operator checks for each statement a whether all total interpretations that can possibly arise from (Q, R) agree on their truth value for the acceptance condition for a. That is, if a has to be accepted no matter how the statements in $S \setminus (Q \cup R)$ are interpreted, then $a \in acc(Q, R)$. The set $rej(Q, R)$ is computed symmetrically, so the pair $(acc(Q, R), rej(Q, R))$ constitutes a refinement of (Q, R).

For $M \subseteq A$, the reduced ADF $D^M = (M, L^M, C^M)$ is defined by $L^M = L \cap M \times M$ and for each $a \in M$ setting $\varphi_a^M = \varphi_a[b/\mathbf{f} : b \notin M]$, that is, replacing all $b \notin M$ by false in the acceptance formula of a. A model M for D is a *stable model* of D iff the least fixpoint of the operator Γ_{D^M} is given by $(M, \emptyset)$. As usual, $su(D)$ and $st(D)$ denote the model sets of the two semantics. While ADF models can be subsets of one another, ADF stable models cannot.

Translations between the formalisms

From AFs to BADFs Brewka and Woltran (2010) showed how to translate AFs into ADFs: For an AF $F = (A, R)$, define the ADF associated to F as $D(F) = (A, R, C)$ with $C = \{\varphi_a\}_{a \in A}$ and $\varphi_a = \bigwedge_{(b,a) \in R} \neg b$ for $a \in A$. Clearly, the resulting ADF is bipolar; parents are always attacking. Brewka and Woltran (2010) proved that this translation is faithful for the AF stable extension and ADF model semantics (Proposition 1). Brewka et al. (2013) later proved the same for the AF stable extension and ADF stable model semantics (Theorem 4). It is easy to see that the translation can be computed in polynomial time.

From ADFs to PL Brewka and Woltran (2010) also showed that ADFs under supported model semantics can be faithfully translated into propositional logic: When acceptance conditions of statements $a \in A$ are represented by propositional formulas φ_a, then the supported models of an ADF D over A are given by the classical models of the formula set $\{a \leftrightarrow \varphi_a \mid a \in A\}$.

From AFs to PL In combination, the previous two translations yield a polynomial and faithful translation chain from AFs into propositional logic.

From ADFs to LPs In recent work we showed that ADFs can be faithfully translated into normal logic programs (Strass, 2013). For an ADF $D = (A, L, C)$, its standard logic program $P(D)$ is given by

$$\{a \leftarrow (M \cup not\,(par(a) \setminus M)) \mid a \in A, C_a(M) = \mathbf{t}\}$$

It is an easy consequence of Lemma 3.14 in (Strass, 2013) that this translation preserves the supported model semantics. For complexity reasons, we cannot expect that this translation is also faithful for the stable semantics. And indeed, the ADF $D = (\{a\}, \{(a, a)\}, \{\varphi_a = a \vee \neg a\})$ has a stable model $\{a\}$ while its standard logic program $P(D) = \{a \leftarrow a, a \leftarrow not\ a\}$ has no stable model.

From AFs to LPs The translation chain from AFs to ADFs to LPs is compact, and faithful for AF stable semantics and LP stable semantics (Osorio et al., 2005), and AF stable semantics and LP supported semantics (Strass, 2013).

From LPs to PL It is well-known that normal logic programs under supported model semantics can be translated to propositional logic (Clark, 1978). There, a logic program P is translated to a propositional theory $\Phi_P = \{a \leftrightarrow \varphi_a \mid a \in A\}$ where

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

for $a \in A$. For the stable model semantics, additional formulas have to be added, but the extended translation works all the same (Lin and Zhao, 2004).

From LPs to ADFs The Clark completion of a normal logic program directly yields an equivalent ADF over the same signature (Brewka and Woltran, 2010). Clearly the translation is computable in polynomial time and the blowup (with respect to the original logic program) is at most linear. The resulting translation is faithful for the supported model semantics, which is a straightforward consequence of Lemma 3.16 in (Strass, 2013).

Relative Expressiveness

We now analyse and compare the relative expressiveness of argumentation frameworks – AFs –, (bipolar) abstract dialectical frameworks – (B)ADFs –, normal logic programs – LPs – and propositional logic – PL. We first look at the different families of semantics – supported and stable models – in isolation and afterwards combine the two. For the languages $L \in \{\text{ADF}, \text{LP}\}$ that have both supported and stable semantics, we will indicate the semantics σ via a superscript as in Definition 1. For AFs we only consider the stable extension semantics, as this is (to date) the only two-valued semantics for AFs. For propositional logic PL we consider the usual model semantics.

With the syntactic translations we reviewed in the previous section, we currently have the following relationships. For the supported semantics,

$$\text{AF} \leq_e \text{BADF}^{su} \leq_e \text{ADF}^{su} \cong_e \text{LP}^{su} \leq_e \text{PL}$$

and for the stable semantics,

$$\text{AF} \leq_e \text{BADF}^{st} \leq_e \text{ADF}^{st} <_e \text{PL}$$
$$\text{AF} \leq_e \text{LP}^{st} <_e \text{PL}$$

Note that $\text{ADF}^{st} <_e \text{PL}$ and $\text{LP}^{st} <_e \text{PL}$ hold since sets of stable models have an antichain property, in contrast to model sets of propositional logic.

Supported semantics

As depicted above, we know that expressiveness from AFs to propositional logic does not decrease. However, it is not yet clear if any of the relationships is strict.

We first show that ADFs can realise any set of models. To show this, we first make a case distinction whether the desired-model set is empty. If there should be no model, we construct an ADF without models. If the set of desired models is nonempty, we construct acceptance conditions directly from the set of desired interpretations. The construction is

similar in design to the one we reviewed for propositional logic, but takes into account the additional interaction between statements and their acceptance conditions.

Theorem 1. *PL* $\leq_e$ *ADF*^{su}

Proof. Consider a vocabulary A and a set $X \subseteq 2^A$. We construct an ADF D_X^{su} with $su(D_X^{su}) = X$ as follows.

1. $X = \emptyset$. We choose some $a \in A$ and set $D_X^{su} = (\{a\}, \{(a,a)\}, \{C_a\})$ with $C_a(\emptyset) = \mathbf{t}$ and $C_a(\{a\}) = \mathbf{f}$. It is easy to see that D_X^{su} has no model.
2. $X \neq \emptyset$. Define $D_X^{su} = (A, L, C)$ where $L = A \times A$ and for each $a \in A$ and $M \subseteq A$, we set $C_a(M) = \mathbf{t}$ iff

$$(M \in X \text{ and } a \in M) \text{ or } (M \notin X \text{ and } a \notin M)$$

We have to show that $M \in X$ iff M is a model for D_X^{su}.

"if": Let M be a model of D_X^{su}.

(a) $M = \emptyset$. Pick any $a \in A$. Since M is a model of D_X^{su}, we have $C_a(M) = \mathbf{f}$. So either (A) $M \in X$ and $a \notin M$ or (B) $M \notin X$ and $a \in M$, by definition of C_a. By assumption $M = \emptyset$, thus $a \notin M$ and $M \in X$.

(b) $M \neq \emptyset$. Let $a \in M$. Then $C_a(M) = \mathbf{t}$ since M is a model of D_X^{su}. By definition of C_a, $M \in X$.

"only if": Let $M \in X$.

(a) $M = \emptyset$. Choose any $a \in A$. By assumption, $a \notin M$ and $M \in X$, whence $C_a(M) = \mathbf{f}$ by definition. Since $a \in A$ was chosen arbitrarily, we have $C_a(M) = \mathbf{f}$ iff $a \notin M$. Thus M is a model of D_X^{su}.

(b) $M \neq \emptyset$. Let $a \in A$. If $a \in M$, then by assumption and definition of C_a we have $C_a(M) = \mathbf{t}$. Conversely, if $a \notin M$, then by definition $C_a(M) = \mathbf{f}$. Since $a \in A$ was arbitrary, M is a model of D_X^{su}. □

When the acceptance conditions are written as propositional formulas, the construction in Theorem 1 simply sets

$$\varphi_a = \bigvee_{M \in X, a \in M} \varphi_M \vee \bigvee_{M \subseteq A, M \notin X, a \notin M} \varphi_M$$

$$\varphi_M = \bigwedge_{a \in M} a \wedge \bigwedge_{a \in A \setminus M} \neg a$$

Since ADFs under supported semantics can be faithfully translated into logic programs, which can be likewise further translated to propositional logic, we have the following.

Corollary 2. *ADF*$^{su} \cong_e$ *LP*$^{su} \cong_e$ *PL*

While general ADFs under the supported model semantics can realise any set of models, the subclass of bipolar ADFs turns out to be less expressive. This is shown using the next result, which allows us to decide realisability of a given model set $X \subseteq 2^A$ in non-deterministic polynomial time. We assume that the size of the input is in the order of $\left|2^A\right|$, that is, the input set X is represented directly. The decision procedure then basically uses the construction of Theorem 1 and an additional encoding of bipolarity to define a reduction to the satisfiability problem in propositional logic.

Theorem 3. *Let $X \subseteq 2^A$ be a set of sets. It is decidable in non-deterministic polynomial time whether there exists a bipolar ADF D with $su(D) = X$.*

Proof. We construct a propositional formula ϕ_X that is satisfiable if and only if X is bipolarly realisable. The propositional signature we use is the following: For each $a \in A$ and $M \subseteq A$, there is a propositional variable p_a^M that expresses whether $C_a(M) = \mathbf{t}$. This allows to encode all possible acceptance conditions for the statements in A. To enforce bipolarity, we use additional variables to model supporting and attacking links: for all $a, b \in A$, there is a variable $p_{sup}^{a,b}$ saying that a supports b, and a variable $p_{att}^{a,b}$ saying that a attacks b. So the vocabulary of ϕ_X is given by

$$P = \left\{ p_a^M, p_{sup}^{a,b}, p_{att}^{a,b} \;\middle|\; M \subseteq A, a \in A, b \in A \right\}$$

To guarantee the desired set of models, we constrain the acceptance conditions as dictated by X: For any desired set M and statement a, the containment of a in M must correspond exactly to whether $C_a(M) = \mathbf{t}$; this is encoded in $\phi_X^{\in}$. Conversely, for any undesired set M and statement a, there must not be any such correspondence, which $\phi_X^{\notin}$ expresses.

$$\phi_X^{\in} = \bigwedge_{M \in X} \left(\bigwedge_{a \in M} p_a^M \wedge \bigwedge_{a \in A \setminus M} \neg p_a^M \right)$$

$$\phi_X^{\notin} = \bigwedge_{M \subseteq A, M \notin X} \left(\bigvee_{a \in M} \neg p_a^M \vee \bigvee_{a \in A \setminus M} p_a^M \right)$$

To enforce bipolarity, we state that each link must be supporting or attacking. To model the meaning of support and attack, we encode all ground instances of their definitions.

$$\phi_{bipolar} = \bigwedge_{a,b \in A} \left(\left(p_{sup}^{a,b} \vee p_{att}^{a,b} \right) \wedge \phi_{sup}^{a,b} \wedge \phi_{att}^{a,b} \right)$$

$$\phi_{sup}^{a,b} = p_{sup}^{a,b} \rightarrow \bigwedge_{M \subseteq A} \left(p_b^M \rightarrow p_b^{M \cup \{a\}} \right)$$

$$\phi_{att}^{a,b} = p_{att}^{a,b} \rightarrow \bigwedge_{M \subseteq A} \left(p_b^{M \cup \{a\}} \rightarrow p_b^M \right)$$

The overall formula is given by $\phi_X = \phi_X^{\in} \wedge \phi_X^{\notin} \wedge \phi_{bipolar}$. The rest of the proof – showing that X is bipolarly realisable if and only if ϕ_X is satisfiable – is delegated to Lemma 12 in the Appendix. □

Remarkably, the decision procedure does not only give an answer, but in the case of a positive answer we can read off the BADF realisation from the satisfying evaluation of the constructed formula. We illustrate the construction with an example that will subsequently be used to show that general ADFs are strictly more expressive than bipolar ADFs.

Example 1. Consider $A = \{x, y, z\}$ and this model set:

$$X_1 = \{\emptyset, \{x, y\}, \{x, z\}, \{y, z\}\}$$

The construction of Theorem 3 yields these formulas:

$$\begin{aligned}\phi^{\in}_{X_1} &= \neg p^{\emptyset}_x \wedge \neg p^{\emptyset}_y \wedge \neg p^{\emptyset}_z \wedge \\ &\quad p^{\{x,y\}}_x \wedge p^{\{x,y\}}_y \wedge \neg p^{\{x,y\}}_z \wedge \\ &\quad p^{\{x,z\}}_x \wedge \neg p^{\{x,z\}}_y \wedge p^{\{x,z\}}_z \wedge \\ &\quad \neg p^{\{y,z\}}_x \wedge p^{\{y,z\}}_y \wedge p^{\{y,z\}}_z \\ \phi^{\notin}_{X_1} &= (\neg p^{\{x\}}_x \vee p^{\{x\}}_y \vee p^{\{x\}}_z) \wedge \\ &\quad (p^{\{y\}}_x \vee \neg p^{\{y\}}_y \vee p^{\{y\}}_z) \wedge \\ &\quad (p^{\{z\}}_x \vee p^{\{z\}}_y \vee \neg p^{\{z\}}_z) \wedge \\ &\quad (\neg p^{\{x,y,z\}}_x \vee \neg p^{\{x,y,z\}}_y \vee \neg p^{\{x,y,z\}}_z)\end{aligned}$$

The remaining formulas about bipolarity are independent of X_1, we do not show them here. We have implemented the translation of Theorem 3 and used the solver clasp (Gebser et al., 2011) to verify that ϕ_{X_1} is unsatisfiable.

A manual proof of bipolar non-realisability of X_1 seems to amount to a laborious case distinction that explores the mutual incompatibility of the disjunctions in $\phi^{\notin}_{X_1}$ and bipolarity, a task that is better left to machines. Together with the straightforward statement of fact that X_1 can be realised by a non-bipolar ADF, the example leads to the next result.

Theorem 4. *BADF^{su} $<_e$ ADF^{su}*

Proof. The model set from Example 1 is realisable under model semantics by ADF D_{X_1} with acceptance conditions

$$\varphi_x = (y \nleftrightarrow z), \quad \varphi_y = (x \nleftrightarrow z), \quad \varphi_z = (x \nleftrightarrow y)$$

where "$\nleftrightarrow$" denotes exclusive disjunction XOR. However, there is no bipolar ADF realising the model set X_1, as is witnessed by unsatisfiability of ϕ_{X_1} and Theorem 3. □

Clearly ADF D_{X_1} is not bipolar since in all acceptance formulas, all statements are neither supporting nor attacking. It is not the only realisation, some alternatives are given by

$$\begin{aligned} D'_{X_1}: &\quad \varphi_x = (y \nleftrightarrow z), \qquad \varphi_y = y, \qquad \varphi_z = z \\ D''_{X_1}: &\quad \varphi_x = x, \qquad \varphi_y = (x \nleftrightarrow z), \qquad \varphi_z = z \\ D'''_{X_1}: &\quad \varphi_x = x, \qquad \varphi_y = y, \qquad \varphi_z = (x \nleftrightarrow y) \end{aligned}$$

This shows that we cannot necessarily use the model set X_1 to determine a *single* reason for bipolar non-realisability, that is, a *single* link (b, a) that is neither supporting nor attacking in *all* realisations. Rather, the culprit(s) might be different in each realisation, and to show bipolar non-realisability, we have to prove that *for all* realisations, there necessarily *exists some* reason for non-bipolarity. And the number of different ADF realisations of a given model set X can be considerable, as our next result shows.

Proposition 5. *Let $|A| = n$, $X \subseteq 2^A$ with $\left|2^A \setminus X\right| = m$. The number of distinct ADFs D with $su(D) = X$ is*

$$r(n, m) = (2^n - 1)^m$$

Proof. We have to count the number of distinct models of the formula $\phi'_X = \phi^{\in}_X \wedge \phi^{\notin}_X$ from the proof of Theorem 3. We first observe that for each $a \in A$ and $M \subseteq A$, the propositional variable p^M_a occurs exactly once in ϕ'_X. Formula $\phi^{\in}_X$ is a conjunction of literals and does not contribute to combinatorial explosion. Formula $\phi^{\notin}_X$ contains m conjuncts. Each of the conjuncts is a disjunction of n distinct literals. There are $2^n - 1$ ways to satisfy such a disjunction. The claim now follows since for each of m conjuncts, we can choose one of $2^n - 1$ different ways to satisfy it. □

So the main contributing factor is the number m of interpretations that are excluded from the desired model set X. For Example 1, for instance, there are $(2^3 - 1)^4 = 7^4 = 2401$ ADFs with the model set X_1. According to Theorem 4, none of them is bipolar. Obviously, the maximal number of realisations is achieved by $X = \emptyset$ whence $r(n, 2^n) = (2^n - 1)^{2^n}$. On the other hand, the model set $X = 2^A$ has exactly one realisation, $r(n, 0) = 1$.

It is comparably easy to show that BADF models are strictly more expressive than AFs, since sets of supported models of bipolar ADFs do not have the antichain property.

Proposition 6. *AF $<_e$ BADF^{su}*

Proof. Consider the vocabulary $A = \{a\}$ and the BADF $D = (A, \{(a, a)\}, \{\varphi_a\})$ with $\varphi_a = a$. It is straightforward to check that its model set is $su(D) = \{\emptyset, \{a\}\}$. Since model sets of AFs under stable extension semantics satisfy the antichain property, there is no equivalent AF over A. □

This yields the following overall relationships:

$$\text{AF} <_e \text{BADF}^{su} <_e \text{ADF}^{su} \cong_e \text{LP}^{su} \cong_e \text{PL}$$

Stable semantics

As before, we recall the current state of knowledge:

$$\text{AF} \leq_e \text{BADF}^{st} \leq_e \text{ADF}^{st} <_e \text{PL} \text{ and } \text{AF} \leq_e \text{LP}^{st} <_e \text{PL}$$

We first show that BADFs are strictly more expressive than AFs.

Proposition 7. *AF $<_e$ BADFst*

Proof. Consider the BADF from Proposition 6, where the acceptance formula of the single statement a is given by $\varphi_a = a$. Its only stable model is $\emptyset$. However there is no AF with a single argument with the same set of stable extensions: the only candidates are $(\{a\}, \emptyset)$ and $(\{a\}, \{(a, a)\})$; their respective stable-extension sets are $\{\{a\}\}$ and $\emptyset$. □

Even if we discount for this special case of realising the empty stable extension, there are non-trivial extension-sets that AFs cannot realise.

Example 2 ((Dunne et al., 2014)). Consider the model set $X_2 = \{\{x, y\}, \{x, z\}, \{y, z\}\}$. Dunne et al. (2014) proved that X_2 is not realisable with stable AF semantics. Intuitively, the argument is as follows: Since x and y occur in an extension together, there can be no attack between them. The same holds for the pairs x, z and y, z. But then the set $\{x, y, z\}$ is conflict-free and thus there must be a stable extension containing all three arguments, which is not allowed by X_2. The reason is AFs' restriction to individual attack, as set attack (also called joint or collective attack) suffices to realise X_2 with BADF D under stable model semantics:

$$\varphi_x = \neg y \vee \neg z, \qquad \varphi_y = \neg x \vee \neg z, \qquad \varphi_z = \neg x \vee \neg y$$

Let us exemplarily show that $M = \{x, y\}$ is a stable model (the other cases are completely symmetric): The reduct D^M is characterised by the two acceptance formulas $\varphi_x = \neg y \vee \neg \mathbf{f}$ and $\varphi_y = \neg x \vee \neg \mathbf{f}$. We then easily find that $\Gamma_{D^M}(\emptyset, \emptyset) = (M, \emptyset) = \Gamma_{D^M}(M, \emptyset)$.

The construction from the previous example model set comes from logic programming (Eiter et al., 2013) and can be generalised to realise any non-empty model set satisfying the antichain property.

Definition 2. Let $X \subseteq 2^A$. Define the following BADF $D^{st}_X = (A, L, C)$ where C_a for $a \in A$ is given by

$$\varphi_a = \bigvee_{M \in X, a \in M} \left(\bigwedge_{b \in A \setminus M} \neg b \right)$$

and thus $L = \{(b, a) \mid M \in X, a \in M, b \in A \setminus M\}$.

We next show that the construction indeed works.

Theorem 8. *Let X with $\emptyset \neq X \subseteq 2^A$ be a $\subseteq$-antichain. We find that $st(D^{st}_X) = X$.*

Proof. Let $M \subseteq A$.

"$\subseteq$": Let $M \notin X$. We show that $M \notin su(D^{st}_X) \supseteq st(D^{st}_X)$.

1. There is an $N \in X$ with $M \subsetneq N$. Then there is an $a \in N \setminus M$. Consider its acceptance formula φ_a. Since $a \in N$ and $N \in X$, the formula φ_a has a disjunct $\psi_{a,N} = \bigwedge_{b \in A \setminus N} \neg b$. Now $M \subseteq N$ implies $A \setminus N \subseteq A \setminus M$ and M is a model for $\psi_{a,N}$. Thus M is a model for φ_a although $a \notin M$, hence $M \notin su(D^{st}_X)$.
2. For all $N \in X$, we have $M \not\subseteq N$. Obviously $M \neq \emptyset$ since $X \neq \emptyset$. Let $a \in M$. For each $N \in X$ with $a \in N$, the acceptance formula φ_a contains a disjunct $\psi_{a,N} = \bigwedge_{b \in A \setminus N} \neg b$. By assumption, for each $N \in X$ there is a $b_N \in M \setminus N$. Clearly $b_N \in A \setminus N$ and b_N is evaluated to true by M. Hence for each $N \in X$ with $a \in N$, the disjunct $\psi_{a,N}$ is evaluated to false by M. Thus φ_a is false under M and $M \notin su(D^{st}_X)$.

"$\supseteq$": Let $M \in X$. We first show that M is a model of D^{st}_X, that is: for all $a \in A$, $a \in M$ iff M is a model for φ_a.

1. Let $a \in M$. By construction, we have that φ_a in D^{st}_X contains a disjunct of the form $\psi_{a,M} = \bigwedge_{b \in A \setminus M} \neg b$. According to the interpretation M, all such $b \in A \setminus M$ are false and thus $\psi_{a,M}$ is true whence φ_a is true.
2. Let $a \in A \setminus M$ and consider its acceptance formula φ_a. Assume to the contrary that M is a model for φ_a. Then there is some $N \in X$ with $a \in N$ such that M is a model for $\psi_{a,N} = \bigwedge_{b \in A \setminus N} \neg b$, that is, $A \setminus N \subseteq A \setminus M$. Hence $M \subseteq N$ and X is not a $\subseteq$-antichain. Contradiction. Thus M is no model for φ_a.

Now consider the reduct D^M of D^{st}_X with respect to M. There, φ^M_a contains the disjunct $\psi^M_{a,M} = \psi_{a,M}[b/\mathbf{f} : b \notin M]$ where all $b \in A \setminus M$ have been replaced by false, whence $\psi^M_{a,M} = \neg \mathbf{f} \wedge \ldots \wedge \neg \mathbf{f}$ and φ^M_a is equivalent to true. Thus each $a \in M$ is true in the least fixpoint of Γ_{D^M} and thus $M \in st(D^{st}_X)$. □

The restriction to non-empty model sets is immaterial, since we can use the construction of Theorem 1 to realise the empty model set.

Since the stable model semantics for both ADFs and normal logic programs have the antichain property, the following is clear.

Corollary 9. *ADFst $\leq_e$ BADFst and LPst $\leq_e$ BADFst*

For the family of stable semantics, this leads to the following overall expressiveness relationships:

$$\text{AF} <_e \text{BADF}^{st} \cong_e \text{ADF}^{st} \cong_e \text{LP}^{st} <_e \text{PL}$$

Supported vs. stable semantics

Now we put the supported and stable pictures together. From the proof of Theorem 8, we can read off that for the canonical realisation D^{st}_X of an antichain X, the supported and stable semantics coincide, that is, $su(D^{st}_X) = st(D^{st}_X) = X$. With this observation, also bipolar ADFs under the supported semantics can realise any antichain, and we have this:

Proposition 10. *BADFst $\leq_e$ BADF^{su}*

As we have seen in Proposition 6, there are bipolar ADFs with supported-model sets that are not antichains. Thus we get the following result.

Corollary 11. *BADFst $<_e$ BADF^{su}*

This result allows us to close the last gap and put together the big picture in Figure 1 below.

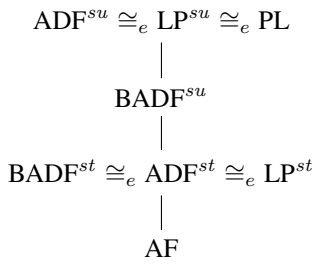

Figure 1: The expressiveness hierarchy. Expressiveness strictly increases from bottom to top. L^σ denotes language L under semantics σ, where "su" is the supported and "st" the stable model semantics; languages are among AFs (argumentation frameworks), ADFs (abstract dialectical frameworks), BADFs (bipolar ADFs), LPs (normal logic programs) and PL (propositional logic).

Discussion

We compared the expressiveness of abstract argumentation frameworks, abstract dialectical frameworks, normal logic programs and propositional logic. We showed that expressiveness under different semantics varies for the formalisms and obtained a neat expressiveness hierarchy. These results inform us about the capabilities of these languages to encode sets of two-valued interpretations, and help us decide which languages to use for specific applications.

For instance, if we wish to encode arbitrary model sets, for example when using model-based revision, then ADFs

and logic programs under supported semantics are a good choice. If we are happy with the restricted class of model sets having the antichain property, then we would be ill-advised to use general ADFs under stable model semantics with their Σ_2^P-hard stable model existence problem; to realise an antichain, it suffices to use bipolar ADFs or normal logic programs, where stable model existence is in NP.

There is much potential for further work. First of all, for results on non-realisability, it would be better to have necessary conditions than having to use a non-deterministic decision procedure. For this, we need to obtain general criteria that all model sets of a given formalism must obey, given the formalism is not universally expressive. This is non-trivial in general, and for AFs it constitutes a major open problem (Dunne et al., 2014; Baumann et al., 2014). Likewise, we sometimes used semantical realisations instead of syntactic ones; for example, to show universal realisability of ADFs under supported models we started out with model sets. It is an interesting question whether a realising ADF can be constructed from a given propositional formula without computing the models of the formula first. Second, there are further semantics for abstract dialectical frameworks whose expressiveness could be studied; Dunne et al. (2014) already analyse many of them for argumentation frameworks. This work is thus only a start and the same can be done for the remaining semantics, for example admissible, complete, preferred and others, which are all defined for AFs, (B)ADFs and LPs (Strass, 2013; Brewka et al., 2013). Third, there are further formalisms in abstract argumentation (Brewka, Polberg, and Woltran, 2013) whose expressiveness is by and large unexplored to the best of our knowledge. Fourth, the requirement that realisations may only use a fixed vocabulary without any additional symbols is quite restrictive. Intuitively, it should be allowed to add a reasonable number of additional atoms, for example a constant number or one that is linear in the original vocabulary. Finally, our study only considered *if* a language can express a model set, but not *to what cost* in terms of representation size. So the natural next step is to consider the succinctness of formalisms, "How large is the smallest knowledge base expressing a given model set?" (Gogic et al., 1995). A landmark result in this direction has been obtained by Lifschitz and Razborov (2006), who have shown that logic programs (with respect to two-valued stable models) are exponentially more succinct than propositional logic. That is, there are logic programs whose respective sets of stable models cannot be expressed by a propositional formula whose size is at most polynomial in the size of the logic program, unless a certain widely believed assumption of complexity theory is false. With the results of the present paper, we have laid the groundwork for a similar analysis of the other knowledge representation languages considered here, perhaps working towards a "map" of these languages in the sense of Darwiche and Marquis' knowledge compilation map [2002].

Acknowledgements. The author wishes to thank Stefan Woltran for providing a useful pointer to related work on realisability in logic programming, and Frank Loebe for several informative discussions. This research was partially supported by DFG (project BR 1817/7-1).

References

Baumann, R.; Dvořák, W.; Linsbichler, T.; Strass, H.; and Woltran, S. 2014. Compact argumentation frameworks. In Konieczny, S., and Tompits, H., eds., *Proceedings of the Fifteenth International Workshop on Non-Monotonic Reasoning (NMR)*.

Bidoit, N., and Froidevaux, C. 1991. Negation by default and unstratifiable logic programs. *Theoretical Computer Science* 78(1):85–112.

Brewka, G., and Woltran, S. 2010. Abstract Dialectical Frameworks. In *Proceedings of the Twelfth International Conference on the Principles of Knowledge Representation and Reasoning (KR)*, 102–111.

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

Brewka, G.; Dunne, P. E.; and Woltran, S. 2011. Relating the Semantics of Abstract Dialectical Frameworks and Standard AFs. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI)*, 780–785. IJCAI/AAAI.

Brewka, G.; Polberg, S.; and Woltran, S. 2013. Generalizations of Dung frameworks and their role in formal argumentation. *IEEE Intelligent Systems* PP(99). Special Issue on Representation and Reasoning. In press.

Clark, K. L. 1978. Negation as Failure. In Gallaire, H., and Minker, J., eds., *Logic and Data Bases*, 293–322. Plenum Press.

Coste-Marquis, S.; Konieczny, S.; Mailly, J.-G.; and Marquis, P. 2013. On the revision of argumentation systems: Minimal change of arguments status. *Proceedings of TAFA*.

Darwiche, A., and Marquis, P. 2002. A Knowledge Compilation Map. *Journal of Artificial Intelligence Research (JAIR)* 17:229–264.

Dimopoulos, Y.; Nebel, B.; and Toni, F. 2002. On the computational complexity of assumption-based argumentation for default reasoning. *Artificial Intelligence* 141(1/2):57–78.

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

Dunne, P. E.; Dvořák, W.; Linsbichler, T.; and Woltran, S. 2014. Characteristics of Multiple Viewpoints in Abstract Argumentation. In *Proceedings of the Fourteenth International Conference on the Principles of Knowledge Representation and Reasoning (KR)*. To appear.

Eiter, T.; Fink, M.; Pührer, J.; Tompits, H.; and Woltran, S. 2013. Model-based recasting in answer-set programming. *Journal of Applied Non-Classical Logics* 23(1–2):75–104.

Gebser, M.; Kaminski, R.; Kaufmann, B.; Ostrowski, M.; Schaub, T.; and Schneider, M. 2011. Potassco: The Potsdam Answer Set Solving Collection. *AI Communications* 24(2):105–124. Available at `http://potassco.sourceforge.net`.

Gelfond, M., and Lifschitz, V. 1988. The Stable Model Semantics for Logic Programming. In *Proceedings of the International Conference on Logic Programming (ICLP)*, 1070–1080. The MIT Press.

Gogic, G.; Kautz, H.; Papadimitriou, C.; and Selman, B. 1995. The comparative linguistics of knowledge representation. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI)*, 862–869. Morgan Kaufmann.

Lifschitz, V., and Razborov, A. 2006. Why are there so many loop formulas? *ACM Transactions on Computational Logic* 7(2):261–268.

Lin, F., and Zhao, Y. 2004. ASSAT: Computing Answer Sets of a Logic Program by SAT Solvers. *Artificial Intelligence* 157(1-2):115–137.

Marek, V. W., and Truszczyński, M. 1991. Autoepistemic logic. *Journal of the ACM* 38(3):587–618.

Osorio, M.; Zepeda, C.; Nieves, J. C.; and Cortés, U. 2005. Inferring acceptable arguments with answer set programming. In *Proceedings of the Sixth Mexican International Conference on Computer Science (ENC)*, 198–205. IEEE Computer Society.

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 the Principles of Knowledge Representation and Reasoning (KR)*. To appear.

Strass, H. 2013. Approximating operators and semantics for abstract dialectical frameworks. *Artificial Intelligence* 205:39–70.

Appendix

Lemma 12. *X is bipolarly realisable if and only if the formula ϕ_X from Theorem 3 is satisfiable.*

Proof. "if": Let $I \subseteq P$ be a model for ϕ_X. For each $a \in A$, we define an acceptance condition as follows: for $M \subseteq A$, set $C_a(M) = \mathbf{t}$ iff $p_a^M \in I$. It is easy to see that $\phi_{bipolar}$ guarantees that these acceptance conditions are all bipolar. The ADF is now given by $D_X^{su} = (A, A \times A, C)$. It remains to show that any $M \subseteq A$ is a model of D_X^{su} if and only if $M \in X$.

"if": Let $M \in X$. We have to show that M is a model of D_X^{su}. Consider any $a \in A$.

1. $a \in M$. Since I is a model of $\phi_X^{\in}$, we have $p_a^M \in I$ and thus by definition $C_a(M) = \mathbf{t}$.
2. $a \in A \setminus M$. Since I is a model of $\phi_X^{\in}$, we have $p_a^M \notin I$ and thus by definition $C_a(M) = \mathbf{f}$.

"only if": Let $M \notin X$. Since I is a model of $\phi_X^{\notin}$, there is an $a \in M$ such that $C_a(M) = \mathbf{f}$ or an $a \notin M$ such that $C_a(M) = \mathbf{t}$. In any case, M is not a model of D_X^{su}.

"only if": Let D be a bipolar ADF with $su(D) = X$. We use D to define a model I for ϕ_X. First, for $M \subseteq A$ and $a \in A$, set $p_a^M \in I$ iff $C_a(M) = \mathbf{t}$. Since D is bipolar, each link is supporting or attacking and for all $a, b \in A$ we can find a valuation for $p_{sup}^{a,b}$ and $p_{att}^{a,b}$. It remains to show that I is a model for ϕ_X.

1. I is a model for $\phi_X^{\in}$: Since D realises X, each $M \in X$ is a model of D and thus for all $a \in A$ we have $C_a(M) = \mathbf{t}$ iff $a \in M$.
2. I is a model for $\phi_X^{\notin}$: Since D realises X, each $M \subseteq A$ with $M \notin X$ is not a model of D. Thus for each such M, there is an $a \in A$ witnessing that M is not a model of D: (1) $a \in M$ and $C_a(M) = \mathbf{f}$, or (2) $a \notin M$ and $C_a(M) = \mathbf{t}$.
3. I is a model for $\phi_{bipolar}$: This is straightforward since D is bipolar by assumption. □