The Relative Expressiveness of Abstract Argumentation and Logic Programming

Hannes Strass
Computer Science Institute
Leipzig University, Germany
strass@informatik.uni-leipzig.de

Abstract

We analyze the relative 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 vocabulary A using only atoms from A. 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. We then demonstrate that this hierarchy collapses once we allow to introduce a linear number of new vocabulary elements.

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 a notion and study the relative expressiveness of argumentation frameworks (AFs) (Dung 1995), normal logic programs (LPs), abstract dialectical frameworks (ADFs) (Brewka and Woltran 2010), and propositional logic (PL).

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 propositional satisfiability problem 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 *realizability*, "What kinds of model sets can the formalism express?"[1]

It is easy to see that propositional logic can express any set of two-valued interpretations, it is *universally expressive*. The same is easy (but less easy) to see for normal logic programs under supported model semantics. For normal logic programs 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. (However, the stable model semantics becomes universally expressive once we allow nested expressions of the form "$not\ not\ p$" in rule bodies (Lifschitz, Tang, and Turner 1999; Lifschitz and Razborov 2006).)

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 the knowledge

[1] In model theory, this is known as *definability*.

base of L_1 to be given in the form of a set X of desired models and construct a semantic *realization* of X in L_2, that is, a knowledge base in L_2 with model set precisely 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 both methods, we can make use of several recent works on the formalisms we study here. First of all, Brewka, Dunne, and Woltran (2011) translated ADFs into AFs for the ADF model and AF stable extension semantics, however this translation introduces additional arguments and is therefore not compact. We (2013) studied the syntactic intertranslatability of ADFs and LPs, but did not look at (B)ADF realizability. Dunne et al. (2014) recently studied realizability for argumentation frameworks. In order to realize a given model set, they allow to introduce any number of new atoms, as long as the new atoms are never true in any model, and presented necessary and sufficient conditions for realizability. There is also recent work by Dyrkolbotn (2014), who analyzed AF realizability under projection (allowing to introduce new atoms) for three-valued semantics, but the results do not apply to our two-valued setting.

The gain that is achieved by our analysis in this paper 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 realizable in the first place. For example, in a recent approach to revising argumentation frameworks (Coste-Marquis et al. 2014), the authors avoid this problem by assuming to produce a *collection* of AFs whose model sets in union produce the desired model set. While the work of Dunne et al. (2014) showed that this is indeed necessary in the case of AFs and stable extension semantics, our work shows that for ADFs under the model semantics, a single knowledge base (ADF) is always enough to realize 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 a polynomial number of new atoms. In theory, an increase from n atoms to n^k atoms for some $k > 1$ is of no consequence. In practice, it has a profound impact: the number n of atoms directly influences the search space that any implementation potentially has to cover. There, the step from 2^n to 2^{n^k} amounts to an *exponential* increase in search space size. Being able to realize 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-realization 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 finally show that allowing to linearly expand the vocabulary leads to a collapse of the hierarchy. The paper concludes with a discussion of possible 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 realizable in propositional logic.[2] This leads us to compare different pairs of languages and semantics with respect to the semantics' range of models. Our concept of "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 set B is called the *body* of the rule, we abbreviate $B^+ = B \cap A$ and $B^- = \{a \in A \mid not\ a \in B\}$. A *logic program (LP)* P over

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

A is a set of logic program rules over A. The body of a rule $a \leftarrow B \in P$ is *satisfied* by a set $M \subseteq A$ iff $B^+ \subseteq M$ and $B^- \cap M = \emptyset$. M is a *supported model* for P iff $M = \{a \in A \mid a \leftarrow B \in P, B \text{ is satisfied by } M\}$. For a logic program P we denote the set of its supported models by $su(P)$. A set $M \subseteq A$ is a *stable model* for P iff M is the $\subseteq$-least supported model of P^M, where 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 (Gelfond and Lifschitz 1988). We write $st(P)$ for the set of stable models of P. It follows from the definition 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 of (abstract) arguments and $R \subseteq A \times A$ a relation of attack between the arguments. 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* is a tuple $D = (A, L, C)$ where A is a set of statements (representing positions in a debate), $L \subseteq A \times A$ is a set of links (representing dependencies between the positions), $C = \{C_a\}_{a \in A}$ is a collection of total functions $C_a : 2^{par(a)} \to \{\mathbf{t}, \mathbf{f}\}$, one for each statement a with direct predecessors (parents) $par(a)$. The function C_a is called *acceptance condition of* a and expresses whether a can be accepted, given the acceptance status of its parents $par(a)$. In this paper, we represent each C_a 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).

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 $\Gamma_D(Q, R) = (ac(Q, R), re(Q, R))$ for $Q, R \subseteq A$, where

$$ac(Q, R) = \{a \in A \mid \forall Z : Q \subseteq Z \subseteq \overline{R} \Rightarrow C_a(Z) = \mathbf{t}\}$$

$$re(Q, R) = \{a \in A \mid \forall Z : Q \subseteq Z \subseteq \overline{R} \Rightarrow C_a(Z) = \mathbf{f}\}$$

(Here, $\overline{R} = A \setminus R$.) 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 $\varphi_a^M = \varphi_a[b/\bot : b \notin M]$, 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 respective model sets; while ADF models can be $\subseteq$-related, 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).

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

From ADFs to LPs We (2013) showed that ADFs can be faithfully translated into normal logic programs. For an ADF $D = (A, L, C)$, its standard LP 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 a consequence of Lemma 3.14 in (Strass 2013) that this translation preserves the supported model semantics.

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 logic programs under supported model semantics can be translated to propositional logic (Clark 1978). A logic program P becomes the 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). The resulting translation is faithful for the supported model semantics, which is a consequence of Lemma 3.16 in (Strass 2013).

Relative Expressiveness

We now analyze 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, $\text{AF} \leq_e \text{LP}^{st} <_e \text{PL}$ and

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

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 does not decrease on the way from AFs up to propositional logic. However, it is not yet clear if any of the relationships is strict.

We first show that ADFs can realize 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 (Footnote 2), but takes into account the additional interaction between statements and their acceptance conditions.

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

Proof. Consider a vocabulary $A \neq \emptyset$ 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 the proof of Theorem 1 defines

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

as acceptance formula of a, where φ_M is as in Footnote 2. 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. *$\text{ADF}^{su} \cong_e \text{LP}^{su} \cong_e \text{PL}$*

While general ADFs under the supported model semantics can realize 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 realizability 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 $su(D) = X$.*

The gist of the proof of Theorem 3 is – given X – to construct a propositional formula $\phi_X = \phi_X^{\in} \wedge \phi_X^{\notin} \wedge \phi_{bipolar}$ that is satisfiable iff X is bipolarly realizable. The vocabulary of ϕ_X contains a propositional variable p_a^M for each $a \in A$ and $M \subseteq A$, where p_a^M expresses whether $C_a(M) = \mathbf{t}$. This allows to encode all possible acceptance conditions for all statements in A; the subformula $\phi_{bipolar}$ ensures bipolarity of the ADF candidates. In consequence, the decision procedure does not only give an answer, but in the case of a positive answer we can read off the BADF realization 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 the vocabulary $A = \{x, y, z\}$ and the model set $X_1 = \{\emptyset, \{x, y\}, \{x, z\}, \{y, z\}\}$. The construc-

tion of Theorem 3 yields $\phi_{X_1} = \phi^{\in}_{X_1} \wedge \phi^{\notin}_{X_1} \wedge \phi_{bipolar}$ with

$$\phi^{\in}_{X_1} = \neg p^{\emptyset}_x \wedge \neg p^{\emptyset}_y \wedge \neg p^{\emptyset}_z \wedge p^{\{x,y\}}_x \wedge p^{\{x,y\}}_y \wedge \neg p^{\{x,y\}}_z \wedge p^{\{x,z\}}_x \wedge \neg p^{\{x,z\}}_y \wedge p^{\{x,z\}}_z \wedge \neg p^{\{y,z\}}_x \wedge p^{\{y,z\}}_y \wedge p^{\{y,z\}}_z$$

$$\phi^{\notin}_{X_1} = (\neg p^{\{x\}}_x \vee p^{\{x\}}_y \vee p^{\{x\}}_z) \wedge (p^{\{y\}}_x \vee \neg p^{\{y\}}_y \vee p^{\{y\}}_z) \wedge (p^{\{z\}}_x \vee p^{\{z\}}_y \vee \neg p^{\{z\}}_z) \wedge (\neg p^{A}_x \vee \neg p^{A}_y \vee \neg p^{A}_z)$$

So $\phi^{\in}_{X_1}$ tells us that $C_x(\emptyset) = \mathbf{f}$, $C_x(\{x,y\}) = \mathbf{t}$, and so on; in contrast, the first conjunct of $\phi^{\notin}_{X_1}$ only tells us that at least one of $C_x(\{x\}) = \mathbf{f}$ or $C_y(\{x\}) = \mathbf{t}$ or $C_z(\{x\}) = \mathbf{t}$ must hold. ($\phi_{bipolar}$ is not shown since it depends only on A.) We implemented the translation and used the solver clasp (Gebser et al. 2011) to verify that ϕ_{X_1} is unsatisfiable.

Together with the straightforward statement of fact that X_1 can be realized by a non-bipolar ADF, Example 1 leads to the next result.

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

Proof. Model set X_1 from Example 1 is realizable under the 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 realizing X_1, as is witnessed by Example 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 comparably easy to show that BADF models are strictly more expressive than AFs, since sets of supported models of bipolar ADFs do not have the antichain property.

Proposition 5. *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 6. *AF $<_e$ BADFst*

Proof. Consider the set $X_2 = \{\{x,y\}, \{x,z\}, \{y,z\}\}$ of desired models. Dunne et al. (2014) proved that X_2 is not realizable with stable AF semantics. However, the model set X_2 is realizable with BADF D_{X_2} under stable 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 characterized by the two acceptance formulas $\varphi_x = \neg y \vee \neg\bot$ and $\varphi_y = \neg x \vee \neg\bot$. We then easily find that $\Gamma_{D^M}(\emptyset,\emptyset) = (M,\emptyset) = \Gamma_{D^M}(M,\emptyset)$. □

The construction that we used in the proof above to realize X_2 comes from logic programming (Eiter et al. 2013) and can be generalized to realize 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 $L = \{(b,a) \mid M \in X, a \in M, b \in A \setminus M\}$.

The next result shows that the construction indeed works.

Theorem 7. *Let X with $\emptyset \neq X \subseteq 2^A$ be a $\subseteq$-antichain. We find that $st(D^{st}_X) = X$.*

The restriction to non-empty model sets is immaterial, since we can use the construction of Theorem 1 to realize the empty model set. As the stable model semantics for ADFs and logic programs both have the antichain property, we get:

Corollary 8. *ADFst $\leq_e$ BADFst and LPst $\leq_e$ BADFst*

This leads to the following overall relationships:

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

We remark that the antichain property provides a *characterization* of realizability with the stable semantics; that is, a model set is stable-realizable iff it is a $\subseteq$-antichain.

Supported vs. stable semantics

Now we put the supported and stable pictures together. It follows from the proof of Theorem 7 that for the canonical realization 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 realize any antichain, and we have this:

Proposition 9. *BADFst $\leq_e$ BADF^{su}*

As we have seen in Proposition 5, there are bipolar ADFs with supported-model sets that are not antichains. We get:

Corollary 10. *BADFst $<_e$ BADF^{su}*

This result allows us to close the last gap and put together the big picture in a Hasse diagram for $\leq_e$:

$$\begin{array}{c} \text{ADF}^{su} \cong_e \text{LP}^{su} \cong_e \text{PL} \\ | \\ \text{BADF}^{su} \\ | \\ \text{BADF}^{st} \cong_e \text{ADF}^{st} \cong_e \text{LP}^{st} \\ | \\ \text{AF} \end{array}$$

Allowing Vocabulary Expansion

Up to here, we only considered *compact* realizations, that do not introduce new vocabulary elements. In this section, we allow the introduction of a small number of new atoms/arguments/statements. More precisely, *small* means the number is linear in the size of the source knowledge base (representing the model set that we wish to realize in a target

language). For the purpose of realizability, the new vocabulary elements are projected out of the resulting models.

As it turns out, adding additional arguments already makes AFs universally expressive (under projection). More technically, we will now show that for each propositional formula φ over vocabulary A, there exists an AF F_φ over an expanded vocabulary $A \cup A_\varphi$ such that the models of φ and the stable extensions of F_φ correspond one-to-one. Roughly, this is possible since AFs can be regarded as a syntactic variant of classical propositional logic that has as its only connective the logical NOR "$\downarrow$" (Gabbay 2011; Brewka, Dunne, and Woltran 2011). Using this connective, negation is expressed by $\neg\varphi \equiv \varphi \downarrow \varphi$ and disjunction by $\varphi \vee \psi \equiv \neg(\varphi \downarrow \psi)$. These equivalences can be used to translate arbitrary propositional formulas (over $\neg, \wedge, \vee$) into the syntactical $\downarrow$-fragment; to guarantee that the size increase is at most linear, we introduce names a_ψ for subformulas ψ. The next definition combines all of these ideas.

Definition 3. Let φ be a formula using $\neg, \wedge, \vee$ over vocabulary A. Define the sets A_φ and R_φ inductively as follows:

$$
\begin{aligned}
A_\top &= \{a_\top\}, A_\bot = \{a_\bot\}, A_p = \{p, a_{\neg p}\} \text{ for } p \in A \\
A_{\neg\xi} &= \{a_{\neg\xi}\} \cup A_\xi \\
A_{\zeta\wedge\xi} &= \{a_{\zeta\wedge\xi}, a_{\neg\zeta}, a_{\neg\xi}\} \cup A_{\neg\zeta} \cup A_{\neg\xi} \\
A_{\zeta\vee\xi} &= \{a_{\zeta\vee\xi}, a_{\zeta\downarrow\xi}\} \cup A_\zeta \cup A_\xi \\
R_\top &= \emptyset, R_\bot = \{(a_\bot, a_\bot)\} \\
R_p &= \{(p, a_{\neg p}), (a_{\neg p}, p)\} \text{ for } p \in A \\
R_{\neg\xi} &= \{(a_\xi, a_{\neg\xi})\} \cup R_\xi \\
R_{\zeta\wedge\xi} &= \{(a_{\neg\zeta}, a_{\zeta\wedge\xi}), (a_{\neg\xi}, a_{\zeta\wedge\xi})\} \cup R_{\neg\zeta} \cup R_{\neg\xi} \\
R_{\zeta\vee\xi} &= \{(a_{\zeta\downarrow\xi}, a_{\zeta\vee\xi}), (a_\zeta, a_{\zeta\downarrow\xi}), (a_\zeta, a_{\zeta\downarrow\xi})\} \cup R_\zeta \cup R_\xi
\end{aligned}
$$

The *AF associated to* φ is given by

$$F_\varphi = (A_\varphi \cup A_\bot, R_\varphi \cup \{(a_\varphi, a_\bot)\} \cup R_\bot)$$

The mutually attacking arguments p and $a_{\neg p}$ for $p \in A$ serve to "guess" a valuation of A, while a_φ and $a_\bot$ guarantee that only (and all) valuations that are models of φ can lead to stable extensions of F_φ: intuitively, $a_\bot$ attacks itself and thus cannot be part of any stable extension; however, it must be attacked, and the only candidate to do so is a_φ.

Our first technical result for this translation shows that the relationships between the newly introduced arguments correctly encode the semantics of the Boolean connectives.

Lemma 11. *Let φ be a formula over vocabulary A and F_φ its associated AF. For each stable extension M of F and $a_\zeta, a_\xi \in A_\varphi$, we have:*

- $a_{\neg\xi} \in M$ *iff* $a_\xi \notin M$;
- $a_{\zeta\wedge\xi} \in M$ *iff both* $a_\zeta \in M$ *and* $a_\xi \in M$;
- $a_{\zeta\vee\xi} \in M$ *iff one of* $a_\zeta \in M$ *or* $a_\xi \in M$;
- $a_{\zeta\downarrow\xi} \in M$ *iff neither* $a_\zeta \in M$ *nor* $a_\xi \in M$.

These correspondences can be used to show by induction that the newly introduced arguments capture the semantics of the formulas they encode (for all subformulas ψ of φ).

Lemma 12. *Let φ be a formula over A and F_φ its associated AF. For each stable extension M of F and $a_\psi \in A_\varphi$, we have $a_\psi \in M$ iff $M \cap A$ is a model of ψ.*

This lets us show the main result of this section, namely that the AF stable extension semantics is universally expressive under projection.

Theorem 13. *Let φ be a formula over vocabulary A and F_φ its associated AF. (1) For each model $M \subseteq A$ of φ, there exists a stable extension E of F_φ with $M \subseteq E$. (2) For each stable extension E of F_φ, the set $E \cap A$ is a model of φ.*

In particular, F_φ has no stable extension iff φ is unsatisfiable. While this shows that the construction of Definition 3 works as intended, it remains to show that the number of new arguments is at most linear in the formula size.

For this, we briefly introduce size measures $\|\cdot\| : L \to \mathbb{N}$ for the two relevant languages L. For propositional logic, $\|\top\| = \|\bot\| = 1 = \|a\|$ for $a \in A$; $\|\neg\varphi\| = \|\varphi\| + 1$; $\|\varphi \wedge \psi\| = \|\varphi \vee \psi\| = \|\varphi\| + \|\psi\| + 1$. For an argumentation framework $F = (A, R)$, define $\|F\| = |A| + |R|$.

We can even show that the total increase in size is only linear, thus also the number of new arguments is linear.

Proposition 14. *For any formula φ, $\|F_\varphi\| \leq 9 \cdot \|\varphi\| + 3$.*

Hence under projection, the AF stable extension semantics can realize as much as propositional logic can. With the results of the previous section (AF $\leq_e$ PL), this means that allowing to introduce a linear number of new vocabulary elements (that are later projected out), all languages considered in this paper are equally (universally) expressive.

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. Furthermore, we have seen that the results are sensitive to the vocabulary one is allowed to use, as the hierarchy collapses when we allow to introduce even only a linear number of new atoms.

There is much potential for further work. First of all, for results on non-realizability, 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 realizations instead of syntactic ones; for example, to show universal realizability of ADFs under supported models we started out with model sets. It is an interesting question whether a realizing ADF can be constructed from a given propositional formula without computing the models of the formula first, just as it is done for AF realization under projection in Definition 3. Second, there are further semantics for abstract dialectical frameworks whose expressiveness could be studied; Dunne et al. (2014) and Dyrkolbotn (2014) already analyze many of them for argumentation frameworks. This work is thus

only a start and the same can be done for the remaining semantics. Third, there are further formalisms in abstract argumentation (Brewka, Polberg, and Woltran 2014) whose expressiveness is by and large unexplored to the best of our knowledge. 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). With the results of the present paper, we have laid important groundwork for a succinctness analysis of the knowledge representation languages considered here.

Acknowledgements. The author wishes to thank Stefan Woltran for providing a useful pointer to related work on realizability 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 *Proceedings of the Twenty-First European Conference on Artificial Intelligence (ECAI)*, 69–74.

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. 2014. Generalizations of Dung frameworks and their role in formal argumentation. *IEEE Intelligent Systems* 29(1):30–38. Special Issue on Representation and Reasoning.

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. 2014. On the revision of argumentation systems: Minimal change of arguments statuses. In *Proceedings of the Fourteenth International Conference on Principles of Knowledge Representation and Reasoning (KR)*, 52–61.

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)*, 72–81.

Dyrkolbotn, S. K. 2014. How to argue for anything: Enforcing arbitrary sets of labellings using AFs. In *Proceedings of the Fourteenth International Conference on the Principles of Knowledge Representation and Reasoning (KR)*, 626–629.

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.

Gabbay, D. M. 2011. Dung's argumentation is essentially equivalent to classical propositional logic with the Peirce-Quine dagger. *Logica Universalis* 5(2):255–318.

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.

Lifschitz, V.; Tang, L.; and Turner, H. 1999. Nested expressions in logic programs. *Annals of Mathematics and Artificial Intelligence* 25(3–4):369–389.

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.

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)*, 101–110.

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