Leipzig University
Faculty of Mathematics and Computer Science
Institute of Computer Science

Augustusplatz 10
04103 Leipzig
Germany

UNIVERSITÄT LEIPZIG

# Approximating Operators and Semantics for Abstract Dialectical Frameworks

Technical Report 1 (2013)

Hannes Strass

ISSN 1430-3701

# Approximating Operators and Semantics for Abstract Dialectical Frameworks

Hannes Strass

Thursday 17 January, 2013

**Abstract**

We provide a systematic in-depth study of the semantics of abstract dialectical frameworks (ADFs), a recent generalisation of Dung's abstract argumentation frameworks. This is done by associating with an ADF its characteristic one-step consequence operator and defining various semantics for ADFs as different fixpoints of this operator. We first show that several existing semantical notions are faithfully captured by our definition, then proceed to define new ADF semantics and show that they are proper generalisations of existing argumentation semantics from the literature. Most remarkably, this operator-based approach allows us to compare ADFs to related nonmonotonic formalisms like Dung argumentation frameworks and propositional logic programs. We use polynomial, faithful and modular translations to relate the formalisms, and our results show that both abstract argumentation frameworks and abstract dialectical frameworks are at most as expressive as propositional normal logic programs.

## 1 Introduction

In recent years, abstract argumentation frameworks (AFs) [9] have become increasingly popular in the artificial intelligence community. An AF can be seen as a directed graph where the nodes are arguments whose internal structure is abstracted away, and where the edges encode a notion of attack between arguments. Part of the reason for the interest in AFs may be that in spite of their conceptual simplicity, there exist many different semantics with different properties in terms of characterisation, existence and uniqueness. Notwithstanding their success, it is yet somewhat unintuitive for users to model argumentation scenarios having as only means of expression arguments *attacking* each other. In legal scenarios, notions of support, joint attack and joint support are very useful, and indeed necessary to model legal proof standards [14].

To overcome the restrictions of Dung's AFs, Brewka and Woltran [3] introduced abstract dialectical frameworks (ADFs), a powerful generalisation of AFs. They showed that ADFs are at least as general as AFs and also provided a (non-modular) translation from normal logic programs to ADFs that preserved stable models. In subsequent work [2], it was shown that ADFs are powerful enough to express argument evaluation structures of the Carneades framework [15], even with cyclic dependencies among arguments. However, the exact location of ADFs in the realm of nonmonotonic knowledge representation formalisms remained unclear.

Later, Brewka et al. [4] were able to give a polynomial translation from ADFs into AFs, suggesting on complexity-theoretical grounds that ADFs are not substantially more expressive than AFs. However, their translation depends on the chosen ADF semantics: one does not simply translate ADFs into AFs with a fixed translation and then gets nice correspondences between the ADF and AF semantics (which is exactly how it works the other way around). Rather, to faithfully map ADFs into AFs one has to decide for a semantics beforehand and then apply a semantics-specific translation. Furthermore, the translation introduced by Brewka et al. [4] for the stable semantics is again not modular, so when something is added to the input ADF, one cannot simply add the translation of the addendum, but has to retranslate the whole updated ADF. In contrast, as we will show, there are translations from AFs and ADFs into normal logic programs (LPs) which are modular, polynomial (in fact linear) and faithful with respect to a whole range of semantics.

These and similar results provide us with a more fine-grained view on the location of AFs and ADFs in the bigger picture of nonmonotonic knowledge representation languages. Technically, we achieve this by a principled and uniform reconstruction of the semantics of abstract dialectical frameworks by embedding them into the approximation operator framework of Denecker, Marek and Truszczyński (henceforth DMT) [6, 7]. In seminal work, DMT developed a powerful algebraic framework in which the semantics of logic programs, default logic and autoepistemic logic can be treated in an entirely uniform and purely algebraic way. The approach works by defining operators, and then their fixpoints according to an abstract and principled method. In this paper, we extend their work by adding abstract dialectical frameworks (and by corollary abstract argumentation frameworks) to their approach.

We do this by defining the so-called *characteristic operator* of an ADF and then deriving new operators following abstract principles [6]. For the special case of a Dung argumentation framework, for instance, the characteristic ADF operator fully captures Dung's characteristic function of the AF. Our investigation generalises the most important semantics known from abstract argumentation to the case of ADFs and relates them to the respective logic programming semantics. It will turn out that when generalising AF semantics, there are typically two different possibilities for generalisations: a "supported" and a "stable" version of the respective semantics. Brewka and Woltran [3] already recognised this in the case of stable extensions for argumentation frameworks: stable AF extensions can be generalised to ADFs in two ways, namely to models and *stable* models for ADFs.

In addition to our usage of operators to clarify the relation of different semantics for *single* formalisms, we will employ another technique to illuminate the relationship between *different* formalisms. This role will be played by investigating polynomial, faithful, modular (PFM) translations between languages as has been done by Gottlob [16] and Janhunen [18] for the relationship between nonmonotonic logics. In our case, we even need a stronger kind of translation: "faithful" usually refers to a translation mapping models of one specific semantics of the source formalism to models of another specific semantics for the target formalism. In our case, faithful refers to the translation providing a perfect alignment with respect to *any* fixpoint semantics or at least a range of fixpoint semantics. Of course, this requires all of the involved semantics to be defined for both source and target formalism, which is however the case for our operator-based approach.

The picture that emerges from our work sheds new light on the underlying connections between the major non-monotonic knowledge representation formalisms, since we study AFs, ADFs and logic programs all in a unified semantical framework. In particular, it shows that Dung's abstract argumentation frameworks can be seen as special cases of propositional normal logic programs. Now all normal logic programs are default theories, which are in turn theories of autoepistemic logic [7]. Thus as a byproduct, our work yields generalisations of argumentation semantics for a general lattice-based setting, from which the existing semantics for logic programming and argumentation can be derived as special cases. Among the semantics generalised are conflict-free and admissible sets, and stage, preferred and semi-stable semantics. As a corollary and another new contribution, this also defines these semantics for default logic and autoepistemic logic [7]. This is a considerable improvement upon a result by Dung [9], who already argued for a preferred semantics for default logic, but only defined it through a translation to infinite argumentation frameworks. We show that our generalisations of argumentation semantics are well-defined by showing that well-known relationships between the semantics generalise accordingly: for example, any preferred ADF model is also complete.

In the last part of the paper, we instantiate the general ADF-based operator to the special case of AFs and present new semantical correspondence results between argumentation frameworks and their translated logic programs: preferred and semi-stable extensions correspond one-to-one to M-stable and L-stable models [21], respectively. Additionally, we show that our lattice-theoretical account of argumentation yields easier proofs for existing results in this area. As our final result, we prove equivalence (in four-valued Belnap logic) of two different translations from AFs to logic programs: a folklore translation from the literature (we call it the standard translation) that encodes attack by negation as failure, and the original translation of Dung [9], where attack and defeat of arguments is explicitly recorded.

**Structure of the paper** We next recall the necessary background, that is to say, the relevant aspects of the DMT lattice-theoretic framework [6, 8], logic programming and argumentation – in particular Dung-style argumentation frameworks and their generalisation to ADFs. Afterwards, we define the characteristic operator of an abstract dialectical framework, whose fixpoints then serve to define ADF semantics in a novel way. The operator will also be used to determine the relationship between propositional normal logic programs and abstract dialectical frameworks: we prove that ADFs can be faithfully and modularly mapped into LPs. We finally show the importance of our general results by illuminating the ramifications for the special case of Dung frameworks. Specifically, we prove several new semantical correspondence results for argumentation and logic programming, and finally prove the equivalence of two different translations from argumentation frameworks into logic programs.

# 2 Background

Let us first recall some basic concepts from lattice theory. A *complete lattice* is a partially ordered set $(L, \sqsubseteq)$ where every subset of $L$ has a least upper and a greatest lower bound. In particular, a complete lattice has a least and a greatest element. An operator $O : L \to L$ is *monotone* if for all $x \sqsubseteq y$ we find $O(x) \sqsubseteq O(y)$; it is *antimonotone* if for all $x \sqsubseteq y$ we find $O(y) \sqsubseteq O(x)$. An $x \in L$ is a *fixpoint* of $O$ if $O(x) = x$; an $x \in L$ is a *prefixpoint* of $O$ if $O(x) \sqsubseteq x$. Due to a fundamental result by Tarski and Knaster, for any monotone operator $O$ on a complete lattice, the set of its fixpoints forms a complete lattice itself [22]. In particular, its least fixpoint $\mathit{lfp}(O)$ exists; additionally, the least prefixpoint of $O$ is also its least fixpoint.

## 2.1 The Algebraic Framework of Denecker et al. [6]

Building upon the fundamental result by Tarski and Knaster, Denecker et al. [6] introduce the important concept of an approximation of an operator. In the study of semantics of nonmonotonic knowledge representation formalisms, elements of lattices represent objects of interest. Operators on lattices transform such objects into others according to the contents of some knowledge base. Consequently, fixpoints of such operators are then objects that cannot be updated any more – informally speaking, the knowledge base can neither add information to a fixpoint nor remove information from it.

To study fixpoints of operators $O$, DMT study fixpoints of their *approximating operators* $\mathcal{O}$.[1] When $O$ operates on a set $L$, its approximation $\mathcal{O}$ operates on pairs $(x, y) \in L^2$ where $L^2$

[1] The approximation of an operator $O$ is typographically indicated by a calligraphic $\mathcal{O}$.

denotes $L \times L$. Such a pair can be seen as representing a *set* of lattice elements by providing a lower bound $x$ and an upper bound $y$. Consequently, the pair $(x, y)$ approximates all $z \in L$ such that $x \sqsubseteq z \sqsubseteq y$. Of special interest are *consistent* pairs – those where $x \sqsubseteq y$, that is, the set of approximated elements is nonempty. A pair $(x, y)$ with $x = y$ is called *exact* – it "approximates" a single element of the original lattice.[2]

There are two natural orderings on approximating pairs: first, the *information ordering* $\leq_i$, that intuitively orders pairs according to their information content. Formally, for $x_1, x_2, y_1, y_2 \in L$ define $(x_1, y_1) \leq_i (x_2, y_2)$ iff $x_1 \sqsubseteq x_2$ and $y_2 \sqsubseteq y_1$. This ordering leads to a complete lattice $(L^2, \leq_i)$, the product of $L$ with itself, its *bilattice*. For example, the pair $(\bot, \top)$ consisting of $\sqsubseteq$-least $\bot$ and $\sqsubseteq$-greatest lattice element $\top$ approximates all lattice elements and thus contains no information – it is the least element of the bilattice $(L^2, \leq_i)$; exact pairs $(x, x)$ are those that are maximally informative while still being consistent. The second natural ordering is the *truth ordering* $\leq_t$, which orders elements of the bilattice according to their degree of truth. Formally, for $x_1, x_2, y_1, y_2 \in L$ it is defined by $(x_1, y_1) \leq_t (x_2, y_2)$ iff $x_1 \sqsubseteq x_2$ and $y_1 \sqsubseteq y_2$. The pair $(\bot, \bot)$ is the least element of $\leq_t$ – in a truth-based setting, it assigns the truth value false to all elements of $L$; the pair $(\top, \top)$ consequently is the $\leq_t$-greatest element – here, all elements of $L$ are assigned value true.

To define an approximation operator $\mathcal{O} : L^2 \to L^2$, one essentially has to define two functions: a function $\mathcal{O}' : L^2 \to L$ that yields a new *lower* bound (first component) for a given pair; and a function $\mathcal{O}'' : L^2 \to L$ that yields a new *upper* bound (second component) for a given pair. Accordingly, the overall approximation is then given by $\mathcal{O}(x, y) = (\mathcal{O}'(x, y), \mathcal{O}''(x, y))$ for $(x, y) \in L^2$. Conversely, in case $\mathcal{O}$ is considered given, the notations $\mathcal{O}'(x, y)$ and $\mathcal{O}''(x, y)$ are read as the projection of $\mathcal{O}(x, y)$ to the first and second component, respectively.

Denecker et al. [6] identify an important subclass of operators on bilattices, namely those that are *symmetric*, that is, for which $\mathcal{O}'(x, y) = \mathcal{O}''(y, x)$. For these, $\mathcal{O}(x, y) = (\mathcal{O}'(x, y), \mathcal{O}'(y, x))$, and to define $\mathcal{O}$ it suffices to specify $\mathcal{O}'$. An operator is *approximating* if it is symmetric and $\leq_i$-monotone. For an antimonotone operator $O$, its *canonical approximation* $\mathcal{O}$ is given by $\mathcal{O}'(x, y) = (O(y), O(x))$.

The main contribution of Denecker et al. [6] was the association of the *stable operator* $\mathcal{SO}$ to an approximating operator $\mathcal{O}$. Below, the expression $\mathcal{O}'(\cdot, y) : L \to L$ denotes the operator given by $x \mapsto \mathcal{O}'(x, y)$ for $x \in L$.

**Definition 2.1.** For a complete lattice $(L, \sqsubseteq)$ and an approximating operator $\mathcal{O} : L^2 \to L^2$, define the

- *complete stable operator for* $\mathcal{O}$ as $c\mathcal{O} : L \to L$ by $c\mathcal{O}(y) \overset{\text{def}}{=} \mathit{lfp}(\mathcal{O}'(\cdot, y))$;
- *stable operator for* $\mathcal{O}$ as $\mathcal{SO} : L^2 \to L^2$ by $\mathcal{SO}(x, y) \overset{\text{def}}{=} (c\mathcal{O}(y), c\mathcal{O}(x))$.

This general, lattice-theoretic definition by DMT yields a uniform treatment of the standard semantics of the major nonmonotonic knowledge representation formalisms – logic programming, default logic and autoepistemic logic [7].

**Definition 2.2.** Let $(L, \sqsubseteq)$ be a complete lattice and $\mathcal{O} : L^2 \to L^2$ be an approximating operator. Furthermore, let $x, y \in L$ with $x \sqsubseteq y$. Define the following semantical notions for $\mathcal{O}$:

[2] Denecker et al. [6] call such pairs "complete," we however use that term for argumentation in a different meaning and want to avoid confusion.

| | |
|---|---|
| Kripke-Kleene semantics | $\mathit{lfp}(\mathcal{O})$ |
| three-valued supported model $(x, y)$ | $\mathcal{O}(x, y) = (x, y)$ |
| two-valued supported model $(x, x)$ | $\mathcal{O}(x, x) = (x, x)$ |
| well-founded semantics | $\mathit{lfp}(\mathcal{SO})$ |
| three-valued stable model $(x, y)$ | $\mathcal{SO}(x, y) = (x, y)$ |
| two-valued stable model $(x, x)$ | $\mathcal{SO}(x, x) = (x, x)$ |

It is clear that each two-valued supported/stable model is a three-valued supported/stable model; furthermore the Kripke-Kleene semantics of an operator is a three-valued supported model and the well-founded semantics is a three-valued stable model. Also, each three-valued/two-valued stable model is a three-valued/two-valued supported model, which is easily seen: if $(x, y)$ is a three-valued stable model, we have $(x, y) = \mathcal{SO}(x, y)$. Now $(x, y) = \mathcal{SO}(x, y) = (c\mathcal{O}(y), c\mathcal{O}(x)) = (\mathit{lfp}(\mathcal{O}'(\cdot, y)), \mathit{lfp}(\mathcal{O}'(\cdot, x)))$ implies $x = \mathcal{O}'(x, y)$ and $y = \mathcal{O}'(y, x)$, whence $(x, y) = (\mathcal{O}'(x, y), \mathcal{O}'(y, x)) = \mathcal{O}(x, y)$ and $(x, y)$ is a three-valued supported model. This holds in particular if $x = y$, and each two-valued stable model is a two-valued supported model.

**Ultimate Approximations** In subsequent work, Denecker et al. [8] presented a general, abstract way to define the most precise approximation of a given operator $O$ in a lattice $(L, \sqsubseteq)$. Most precise here refers to a generalisation of $\leq_i$ to operators, where for $\mathcal{O}_1, \mathcal{O}_2 : L^2 \to L^2$, they define $\mathcal{O}_1 \leq_i \mathcal{O}_2$ iff for all $x \sqsubseteq y \in L$ it holds that $\mathcal{O}_1(x, y) \leq_i \mathcal{O}_2(x, y)$. For consistent pairs $(x, y)$ of the bilattice $(L^2, \leq_i)$, they show that the most precise – called the *ultimate* – approximation of $O$ is given by $\mathcal{U}_O(x, y) \overset{\text{def}}{=} (\mathcal{U}'_O(x, y), \mathcal{U}''_O(x, y))$ with

$$\mathcal{U}'_O(x, y) \overset{\text{def}}{=} \bigsqcap \{O(z) \mid x \sqsubseteq z \sqsubseteq y\}$$
$$\mathcal{U}''_O(x, y) \overset{\text{def}}{=} \bigsqcup \{O(z) \mid x \sqsubseteq z \sqsubseteq y\}$$

Note that the ultimate approximation works only for consistent pairs and is not symmetric. Still, this definition is remarkable since previously, approximating operators $\mathcal{O}$ for lattice operators $O$ had to be devised by hand rather than automatically derived. We next illustrate the workings of the operator-based framework for the case of logic programming.

## 2.2 Logic Programming

For technical convenience, we use definitions along the lines of Fitting [12], whose fixpoint-theoretic approach to logic programming was extended by Denecker et al. [6]. For a nonempty set $A$ – the *signature*, or set of *atoms* –, define $\mathit{not}\, A \overset{\text{def}}{=} \{\mathit{not}\, a \mid a \in A\}$ and the set of *literals* over $A$ as $\mathit{Lit}(A) \overset{\text{def}}{=} A \cup \mathit{not}\, A$. A *logic program rule* over $A$ is then of the form $a \leftarrow M$ where $a \in A$ and $M \subseteq \mathit{Lit}(A)$. The rule can be read as logical consequence, "$a$ is true if all literals in $M$ are true." We denote by $M^+ \overset{\text{def}}{=} M \cap A$ and $M^- \overset{\text{def}}{=} \{a \in A \mid \mathit{not}\, a \in M\}$ the *positive* and *negative body* atoms, respectively. A rule is *definite* if $M^- = \emptyset$. For singleton $M = \{m\}$ we denote the rule just by $a \leftarrow m$. A *logic program (LP)* $\Pi$ over $A$ is a set of logic program rules over $A$, and it is definite if all rules in it are definite.

The perhaps most prominent example for an operator is the one-step consequence operator $T_\Pi$ associated with a definite logic program $\Pi$ [12]. For a signature $A$, it operates on subsets of $A$ and assigns to a set of atoms $S$ those atoms which are implied by $S$ according to the rules in $\Pi$. The underlying lattice is therefore $(2^A, \subseteq)$ consisting of the set of $A$'s subsets ordered by $\subseteq$.

This operator was later generalised to four-valued Belnap logic [12] and can be recast in a bilattice-based setting as follows. A pair $(X, Y) \in 2^A \times 2^A$ can be read as a four-valued assignment by evaluating all atoms in $X \cap Y$ as true, those in $A \setminus (X \cup Y)$ as false, the ones in $Y \setminus X$ as undefined and the atoms in $X \setminus Y$ as inconsistent.

**Definition 2.3.** For a logic program $\Pi$ over $A$, define an (approximating) operator $\mathcal{T}_\Pi : 2^A \times 2^A \to 2^A \times 2^A$ as follows: for $X, Y \subseteq A$,

$$\mathcal{T}_\Pi(X,Y) \overset{\text{def}}{=} (\mathcal{T}'_\Pi(X,Y), \mathcal{T}'_\Pi(Y,X))$$
$$\mathcal{T}'_\Pi(X,Y) \overset{\text{def}}{=} \left\{ a \in A \;\middle|\; a \leftarrow M \in \Pi, M^+ \subseteq X, M^- \cap Y = \emptyset \right\}$$

Roughly, to construct a new lower bound, the operator $\mathcal{T}'_\Pi$ returns all those atoms for which a rule exists whose positive body is implied by the current lower bound and whose negative body does not share an atom with the current upper bound. This first of all means that the operator allows to infer an atom via a program rule if – according to the input estimate – the positive body is true and the negative body is false. The fixpoints of $\mathcal{T}_\Pi$ are the four-valued *supported models* of $\Pi$; its consistent fixpoints are the three-valued supported models of $\Pi$. The two-valued supported models of $\Pi$ are computed by the abovementioned operator $T_\Pi$, that – in this setting – is defined by $T_\Pi(M) = \mathcal{T}'_\Pi(M, M)$ [6].

The abstract principles of Denecker et al. [6] outlined above also yield the corresponding *stable* operator $\mathcal{ST}_\Pi$. This operator in turn immediately yields the Gelfond-Lifschitz operator $GL_\Pi(M) = \mathcal{ST}'_\Pi(M, M)$ for computing two-valued stable models of $\Pi$. The stable operator $\mathcal{ST}_\Pi$ also gives rise to the *well-founded model* of $\Pi$, which is the least fixpoint of $\mathcal{ST}_\Pi$. Additionally, *three-valued stable models* are the consistent fixpoints of $\mathcal{ST}_\Pi$. These are further refined into two additional semantics: *M-stable models* are three-valued stable models $(X, Y)$ where $X$ is $\subseteq$-maximal – M-stable is for "maximal stable" [21]; *L-stable models* are three-valued stable models $(X, Y)$ where $Y \setminus X$ is $\subseteq$-minimal – L-stable is for "least undefined" [21]. It is clear that these same maximisation/minimisation criteria can be applied to consistent fixpoints of $\mathcal{T}_\Pi$ – the three-valued supported models. This leads to *M-supported models* and *L-supported models.* In a table much like the one from Definition 2.2, this looks thus:

| | |
|---|---|
| M-supported model $(X, Y)$ | $\mathcal{T}_\Pi(X, Y) = (X, Y)$ and $(X, Y)$ is $\leq_i$-maximal |
| L-supported model $(X, Y)$ | $\mathcal{T}_\Pi(X, Y) = (X, Y)$ and $Y \setminus X$ is $\subseteq$-minimal |
| M-stable model $(X, Y)$ | $\mathcal{ST}_\Pi(X, Y) = (X, Y)$ and $(X, Y)$ is $\leq_i$-maximal |
| L-stable model $(X, Y)$ | $\mathcal{ST}_\Pi(X, Y) = (X, Y)$ and $Y \setminus X$ is $\subseteq$-minimal |

It follows that each two-valued supported/stable model is an L-supported/L-stable model is an M-supported/M-stable model is a three-valued supported/stable model.

As an example, consider the logic program $\pi_1 = \{a \leftarrow \emptyset, b \leftarrow a\}$. It is a definite LP, thus we can iterate its two-valued one-step consequence operator $T_{\pi_1}$ on the empty set, the least element of the relevant lattice: we have $T_{\pi_1}(\emptyset) = \{a\}$ and $T_{\pi_1}(\{a\}) = \{a, b\} = T_{\pi_1}(\{a, b\})$ as a fixpoint and thus the least (two-valued supported) model of program $\pi_1$. Now we add another rule to this program and set $\pi_2 \overset{\text{def}}{=} \pi_1 \cup \{c \leftarrow \{b, \mathit{not}\, d\}\}$, a logic program over $A = \{a, b, c, d\}$ that is not definite. To compute its well-founded model, we iterate the associated stable four-valued one-step consequence operator $\mathcal{ST}_{\pi_2}$ on the least element $(\emptyset, A)$ of the relevant bilattice. We see that $\mathcal{ST}_{\pi_2}(\emptyset, A) = (\{a\}, \{a, b, c\})$: intuitively, $a$ is added to the lower bound since its body is satisfied, $d$ is removed from the upper bound because there is no program rule to derive $d$. Applying $\mathcal{ST}_{\pi_2}$ again leads to the pair $(\{a, b, c\}, \{a, b, c\})$ which is an exact fixpoint and thus the only two-valued stable model of $\pi_2$.

## 2.3 Abstract Argumentation

Dung [9] introduced a way to study the fundamental mechanisms that humans use in argumentation. His argumentation frameworks (AFs) $\Theta$ are pairs $(A, R)$ where $A$ is a set and $R \subseteq A \times A$. The intended reading of an AF $\Theta$ 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: for $a, b \in A$ a pair $(a, b) \in R$ expresses that argument $a$ attacks argument $b$ in some sense. This seemingly lightweight formalism allows for a rich semantical theory, whose most important notions we subsequently recall.

The purpose of semantics for argumentation frameworks is to determine sets of arguments which are acceptable according to various standards. As an intuitive example, a set of arguments could be accepted if it is internally consistent and can defend itself against attacks from the outside. More formally, a set $S \subseteq A$ of arguments is *conflict-free* iff there are no $a, b \in S$ with $(a, b) \in R$. For an argument $a \in A$, the set of its attackers is $\mathit{Attackers}_\Theta(a) \overset{\text{def}}{=} \{b \in A \mid (b, a) \in R\}$. An AF is *finitary* iff $\mathit{Attackers}_\Theta(a)$ is finite for all $a \in A$. For $S \subseteq A$, the set of arguments it attacks is $\mathit{Attacked}_\Theta(S) \overset{\text{def}}{=} \{b \in A \mid (a, b) \in R \text{ for some } a \in S\}$. Finally, for $S \subseteq A$ and $a \in A$, the set $S$ *defends* $a$ iff $\mathit{Attackers}_\Theta(a) \subseteq \mathit{Attacked}_\Theta(S)$, that is, all attackers of $a$ are attacked by $S$.

The major semantics for argumentation frameworks can be formulated using two operators that Dung [9] already studied. The first is the *characteristic function* of an AF $\Theta = (A, R)$: for $S \subseteq A$, define $F_\Theta(S) \overset{\text{def}}{=} \{a \in A \mid S \text{ defends } a\}$. This operator $F_\Theta$ is $\subseteq$-monotone and therefore has a least fixpoint in the lattice $(2^A, \subseteq)$. This least fixpoint of $F_\Theta$ is defined as the *grounded* extension of $\Theta$. The second relevant operator $U_\Theta$ takes as input a set $S$ of arguments, and returns the arguments which are not attacked by any argument in $S$ ($U$. is for "unattacked"). It is an antimonotone operator, and its fixpoints are the *stable* extensions of $\Theta$. Additionally, $U_\Theta$ can characterise conflict-freeness: a set $S \subseteq A$ is conflict-free iff $S \subseteq U_\Theta(S)$. Further semantics are defined as follows. A set $E \subseteq A$ is a *complete* extension iff it is a conflict-free fixpoint of $F_\Theta$. More generally, a set $S \subseteq A$ is *admissible* iff $S$ is conflict-free and $S \subseteq F_\Theta(S)$. A set $S \subseteq A$ is a *stage* extension iff it is conflict-free and $S \cup \mathit{Attacked}_\Theta(S)$ is $\subseteq$-maximal. Finally, *preferred* extensions are $\subseteq$-maximal complete extensions; and *semi-stable* extensions are those complete extensions $E$ where $E \cup \mathit{Attacked}_\Theta(E)$ is $\subseteq$-maximal. For two argumentation frameworks $\Theta_1 = (A_1, R_1)$ and $\Theta_2 = (A_2, R_2)$, their union is defined as $\Theta_1 \cup \Theta_2 \overset{\text{def}}{=} (A_1 \cup A_2, R_1 \cup R_2)$.

As an example, let the argumentation framework $\theta = (A, R)$ be given by $A = \{a, b, c, d\}$ and $R = \{(a, b), (c, d), (d, c)\}$. It is depicted by the following directed graph:

Its grounded extension is the set $G = \{a\}$; it possesses two stable extensions, $E_1 = \{a, c\}$ and $E_2 = \{a, d\}$. The three sets $G, E_1, E_2$ form the only complete extensions of $\theta$.

## 2.4 Abstract Dialectical Frameworks

Brewka and Woltran [3] introduced abstract dialectical frameworks as a powerful generalisation of abstract argumentation frameworks that are able to capture not only attack and support, but also more general notions such as joint attack and joint support.

**Definition 2.4.** An *abstract dialectical framework (ADF)* is a triple $\Xi = (S, L, C)$ where

- $S$ is a set of *statements*,
- $L \subseteq S \times S$ is a set of *links*, where $\mathit{par}(s) \overset{\text{def}}{=} \{r \in S \mid (r, s) \in L\}$
- $C = \{C_s\}_{s \in S}$ is a set of total functions $C_s : 2^{\mathit{par}(s)} \to \{\mathit{in}, \mathit{out}\}$.

Intuitively, the function $C_s$ for a statement $s$ determines the acceptance status of $s$, which naturally depends on the status of its parent nodes. Alternatively, any such function $C_s$ can be represented by a set $C_s^{in} \overset{\text{def}}{=} \{M \subseteq \mathit{par}(s) \mid C_s(M) = \mathit{in}\}$. We will use both representations in this paper and indicate the alternative one by writing an ADF as $(S, L, C^{in})$. A third alternative representation of an acceptance condition $C_a$ (also introduced by Brewka and Woltran [3]) is a propositional formula $\varphi_a$ over the vocabulary $par(a)$. The understanding here is that $C_a^{in}$ is given by the two-valued models of $\varphi_a$, where an interpretation is identified with the set of atoms that are evaluated to true.

**Example 2.1.** The following is a simple ADF: $D = (S, L, C^{in})$ with statements $S = \{a, b, c, d\}$, links $L = \{(a, c), (b, b), (b, c), (b, d)\}$ and acceptance functions given by $C_a^{in} = \{\emptyset\}$, $C_b^{in} = \{\{b\}\}$, $C_c^{in} = \{\{a, b\}\}$ and $C_d^{in} = \{\emptyset\}$. These acceptance functions can intuitively be interpreted as follows:

- Statement $a$ has no parents, $par(a) = \emptyset$, thus $2^{par(a)} = \{\emptyset\}$. The acceptance function specifies that $\emptyset \mapsto in$, whence $a$ is always *in*.

- Statement $b$ is its own parent. According to its acceptance function, it is *in* only if it is *in*. Statement $b$ is thus (cyclicly) self-supporting.

- Statement $c$ has parents $par(c) = \{a, b\}$. They jointly support $c$, as is witnessed by $C_c^{in} = \{par(c)\}$. Note that joint support here indeed means that the support only becomes effective if *both* parents are *in*.

- Statement $d$ is attacked by its only parent $b$.

Brewka and Woltran [3] introduced several semantical notions for ADFs. For an ADF $\Xi = (S, L, C^{in})$, a set $M \subseteq S$ is *conflict-free* iff for all $s \in M$ we have $M \cap par(s) \in C_s^{in}$. A set $M \subseteq S$ is a *model for* $\Xi$ iff for each $s \in S$ we have $s \in M$ iff $M \cap par(s) \in C_s^{in}$.

**Example 2.1** (Continued)**.** A conflict in a set of statements intuitively means that there is either an attack within the set or a lack of support for some statement. The running example ADF $D$ has the following conflict-free sets:

$$\emptyset, \{a\}, \{b\}, \{d\}, \{a, b\}, \{a, d\}, \{a, b, c\}$$

This is easy to understand: from all subsets of $S = \{a, b, c, d\}$, we have to remove those that (1) contain both $b$ and $d$, since $b$ attacks $d$; or (2) contain $c$ without containing both $a$ and $b$, because $c$ depends on joint support of $a$ and $b$. The remaining ones above are conflict-free.

The two models of $D$ are $\{a, b, c\}$ and $\{a, d\}$.

Some semantics were only defined for a subclass of ADFs called *bipolar*. Intuitively, in bipolar ADFs (BADFs) each link is supporting or attacking (or both); that is, there is nothing such as joint support or attack and the like. Formally, a link $(r, s) \in L$ is *supporting in* $\Xi$ iff for all $R \subseteq par(s)$, we have that $R \in C_s^{in}$ implies $R \cup \{r\} \in C_s^{in}$; symmetrically, a link $(r, s) \in L$ is *attacking in* $\Xi$ iff for all $R \subseteq par(s)$, we have that $R \cup \{r\} \in C_s^{in}$ implies $R \in C_s^{in}$. An ADF $\Xi = (S, L, C)$ is *bipolar* iff all links in $L$ are either supporting or attacking; we use $L^+$ to denote all supporting and $L^-$ to denote all attacking links of $L$ in $\Xi$. A model $M$ of a bipolar ADF $\Xi$ is a *BW-stable model of* $\Xi$ iff it is the least model of the reduced ADF $\Xi^M$ defined as $\Xi^M = (S^M, L^M, C^M)$ with

- $S^M = S \cap M$ (nodes are restricted to those in the model),

- $L^M = \left\{(r, s) \;\middle|\; r, s \in S^M, (r, s) \in L^+\right\}$ (links are restricted to supporting links among nodes in the model) and

- for each $s \in S^M$ and $B \subseteq S^M$, we set $C_s^M(B) = in$ iff $C_s(B) = in$ (likewise the acceptance functions are restricted to the remaining parent nodes).

Stable models then serve to define further notions; but first let us define how to remove a set $R$ of statements from an ADF $\Xi = (S, L, C^{in})$ as follows. $\Xi - R \stackrel{\text{def}}{=} (S', L', C')$, where

- $S' = S \setminus R$ (the nodes in $R$ are removed),
- $L' = L \cap (S' \times S')$ (links are restricted to the remaining nodes) and
- $C' = \left\{\left\{B \cap S' \;\middle|\; B \in C_s^{in}\right\}\right\}_{s \in S'}$ (likewise, acceptance conditions are restricted to the remaining parents).

For a bipolar ADF $\Xi = (S, L, C)$, a set $M \subseteq S$ is *BW-admissible in* $\Xi$ iff there is some $R \subseteq S$ with

- $L^- \cap (R \times M) = \emptyset$ (there are no attacks from $R$ to $M$) and
- $M$ is a stable model of $\Xi - R$.

A set $M \subseteq S$ is a *BW-preferred model of* $\Xi$ iff it is $\subseteq$-maximal among the sets BW-admissible in $\Xi$. Finally, Brewka and Woltran [3] also generalise the grounded semantics: for $\Xi = (S, L, C)$ they define a monotone operator $\Gamma_\Xi : 2^S \times 2^S \to 2^S \times 2^S$ by $(X, Y) \mapsto (\Gamma'_\Xi(X, Y), \Gamma''_\Xi(X, Y))$, where[3]

$$\Gamma'_\Xi(X, Y) \stackrel{\text{def}}{=} \left\{s \in S \;\middle|\; \text{for all } X \subseteq Z \subseteq Y, \text{we have } Z \cap par(s) \in C_s^{in}\right\}$$
$$\Gamma''_\Xi(X, Y) \stackrel{\text{def}}{=} \left\{s \in S \;\middle|\; \text{there exists } X \subseteq Z \subseteq Y \text{ with } Z \cap par(s) \in C_s^{in}\right\}$$

The $\leq_i$-least fixpoint of $\Gamma_\Xi$ gives rise to the *BW-well-founded model of* $\Xi$.

**Example 2.1** (Continued)**.** The $\leq_i$-least fixpoint of $\Gamma_D$ is the pair $(\{a\}, \{a, b, c, d\})$, therefore the BW-well-founded model of $D$ is the set $\{a\}$. Intuitively, statement $a$ is in there because it is always *in*. Statement $b$ is not contained in the BW-well-founded model since it is only self-supporting. Statement $c$ is not contained because it needs joint support by $a$ and $b$, of which $b$ is missing. For $d$, it cannot be guaranteed that its attacker $b$ is necessarily *out*, since it is still contained in the upper bound of $\Gamma_D$'s least fixpoint.

It is clear that ADFs are a generalisation of AFs: for an argumentation framework $\Theta = (A, R)$, its *associated abstract dialectical framework* is $\Xi(\Theta) = (A, R, C^{in})$, where $C_a^{in} = \{\emptyset\}$ for each $a \in A$. But this is not just syntactical; Brewka and Woltran [3] showed that their semantical notions for ADFs are generalisations of Dung's respective AF notions:

**Proposition 2.1.** *Let $\Theta = (A, R)$ be an argumentation framework and $\Xi(\Theta) = (A, R, C^{in})$ its associated abstract dialectical framework. The following are in one-to-one correspondence:*

1. *the grounded extension of $\Theta$ and the BW-well-founded model of $\Xi(\Theta)$;*
2. *conflict-free sets of $\Theta$ and conflict-free sets of $\Xi(\Theta)$;*
3. *stable extensions of $\Theta$ and models of $\Xi(\Theta)$;*
4. *stable extensions of $\Theta$ and BW-stable models of $\Xi(\Theta)$;*
5. *preferred extensions of $\Theta$ and BW-preferred models of $\Xi(\Theta)$.*

*Proof.* Propositions 3, 1, 7 and 12 of [3]. □

It is especially notable that models and stable models coincide for AF-based ADFs, a fact that we will illuminate further and be able to provide an intuitive explanation for.

[3] The representation of the operator and the lattice it operates on given by Brewka and Woltran [3] is slightly different: both representations use pairs of sets of statements to describe the current acceptance status of statements. Their pairs explicitly represent the statements that are *in* in the first component and the ones that are *out* in the second component. Since our second component explicitly represents the statements that are *not out*, we adjusted the definition of the operator $\Gamma''_\Xi$ for computing the second component.

# 3 Approximating Semantics of Abstract Dialectical Frameworks

Abstract dialectical frameworks are nonmonotonic knowledge representation formalisms. As such, they allow to express knowledge and provide formal semantics for such expressions. In this respect, nonmonotonic means that extending a knowledge base (that is, an ADF) may invalidate conclusions drawn from it. One approach to define semantics for knowledge bases is the one championed by van Emden, Kowalski and others: there, a revision operator is associated with a knowledge base [12]. The operator revises interpretations for the knowledge base $K$ in the sense that the revision of an interpretation is somehow "more in accord" with the knowledge contained in $K$. Extending the metaphor, fixpoints of the revision operator then correspond to models since they exactly "hit the spot" in that they represent stationary interpretations that cannot be revised further. In this section, we will apply this operator-based approach to semantics to abstract dialectical frameworks.

From the definition of a model of an ADF by Brewka and Woltran [3], it is straightforward to devise a two-valued one-step consequence operator for a given ADF: given a two-valued interpretation, we evaluate the acceptance condition of each statement; the resulting evaluation determines the revised interpretation. To generalise this to an approximating operator, we generalise the evaluation to four-valued Belnap logic.

## 3.1 The Characteristic Operator of an ADF

For an abstract dialectical framework $\Xi = (S, L, C^{in})$, four-valued interpretations can be represented by pairs $(X, Y)$ with $X, Y \subseteq S$. Such pairs can equivalently be interpreted as approximations to two-valued interpretations where $X$ represents a lower bound and $Y$ an upper bound of the approximation. Given such an approximating pair $(X, Y)$ and an ADF $\Xi$, to revise the pair we do the following for each statement $s \in S$: we check if there is some subset $B$ of the parents of $s$ (which are exactly the statements that determine the acceptance status of $s$) such that (1) all statements in $B$ being *in* causes $s$ to be *in*; (2) all statements in $B$ are indeed *in* according to the conservative estimate $X$; (3) the remaining parents of $s$ are indeed *out*, that is, not contained in the liberal estimate $Y$. The definition below, the most important definition of the paper, makes this formally precise.

**Definition 3.1.** Let $\Xi = (S, L, C^{in})$ be an abstract dialectical framework. Define an operator $\mathcal{G}_\Xi : 2^S \times 2^S \to 2^S \times 2^S$ by

$$\mathcal{G}_\Xi(X, Y) \overset{\text{def}}{=} (\mathcal{G}'_\Xi(X, Y), \mathcal{G}'_\Xi(Y, X))$$
$$\mathcal{G}'_\Xi(X, Y) \overset{\text{def}}{=} \left\{ s \in S \;\middle|\; B \in C^{in}_s, B \subseteq X, (par(s) \setminus B) \cap Y = \emptyset \right\}$$

A two-valued immediate consequence operator for ADFs (the equivalent of logic programs' two-valued van Emden-Kowalski operator $T_\Pi$) is now given by $G_\Xi(X) \overset{\text{def}}{=} \mathcal{G}'_\Xi(X, X)$. The next lemma about this two-valued operator relates to ADF models and will prove useful on various occasions.

**Lemma 3.1.** *For any abstract dialectical framework $\Xi = (S, L, C)$, $s \in S$ and $X \subseteq S$ we have $s \in G_\Xi(X)$ iff $X \cap par(s) \in C^{in}_s$.*

*Proof.*

$$\begin{aligned} s \in G_\Xi(X) &\text{ iff } s \in \mathcal{G}'_\Xi(X, X) \\ &\text{ iff } X' \in C_s^{in}, X' \subseteq X, (par(s) \setminus X') \cap X = \emptyset, X \cap par(s) = X' \\ &\text{ iff } X \cap par(s) \in C_s^{in} \end{aligned}$$ □

Our definition of the approximating operator of an ADF immediately defines quite a number of semantics for ADFs, among them all the semantics of Definition 2.2. In the following, we will show how some of the standard operator-based semantics coincide with existing ADF semantics. Operator-based semantics without a corresponding ADF semantics accordingly define new semantical notions for abstract dialectical frameworks, for example three-valued stable models. Similarly, there are ADF semantics which have no operator-based counterpart – BW-stable, BW-admissible and BW-preferred –, we will provide alternative, operator-based definitions for these semantics.

But first, we do the obviously necessary and show that $\mathcal{G}_\Xi$ is indeed an approximating operator. From Definition 3.1 it is immediate that $\mathcal{G}_\Xi$ is symmetric. It is easy to prove that the operator is also $\leq_i$-monotone.

**Proposition 3.2.** *For any ADF $\Xi = (S, L, C)$, the operator $\mathcal{G}_\Xi$ is $\leq_i$-monotone.*

*Proof.* Let $(X_1, Y_1) \leq_i (X_2, Y_2)$, that is, $X_1 \subseteq X_2$ and $Y_2 \subseteq Y_1$. We have to show $\mathcal{G}_\Xi(X_1, Y_1) \leq_i \mathcal{G}_\Xi(X_2, Y_2)$, that is, (1) $\mathcal{G}'_\Xi(X_1, Y_1) \subseteq \mathcal{G}'_\Xi(X_2, Y_2)$ and (2) $\mathcal{G}'_\Xi(Y_2, X_2) \subseteq \mathcal{G}'_\Xi(Y_1, X_1)$.

1. Let $s \in \mathcal{G}'_\Xi(X_1, Y_1)$. Then there is an $M \in C_s^{in}$ with $M \subseteq X_1$ and $(par(s) \setminus M) \cap Y_1 = \emptyset$. Now $M \subseteq X_1 \subseteq X_2$; furthermore $Y_2 \subseteq Y_1$ implies $(par(s) \setminus M) \cap Y_2 = \emptyset$, whence $s \in \mathcal{G}'_\Xi(X_2, Y_2)$.
2. Analogous. □

Hence the fixpoints of this operator form a complete lattice [22]. From $\mathcal{G}_\Xi$ being approximating it follows that it maps consistent pairs to consistent pairs [6, Proposition 14]; in particular its least fixpoint is consistent. Finally, we can construct its associated stable operator $\mathcal{SG}_\Xi$ as defined by Denecker et al. [6]. We will now use our newly defined approximating ADF operator to systematically reconstruct semantical notions for abstract dialectical frameworks.

### 3.1.1 Conflict-free sets

First of all, we find a nice characterisation of conflict-freeness: a set is conflict-free iff application of the approximating operator leads to a pair which is at least as high up in the truth value ordering $\leq_t$. Informally speaking, applying the approximating operator to a conflict-free pair keeps or increases the truth value of the pair.

**Proposition 3.3.** *For any abstract dialectical framework $\Xi = (S, L, C)$, a set $M \subseteq S$ is conflict-free for $\Xi$ iff $(M, M) \leq_t \mathcal{G}_\Xi(M, M)$.*

*Proof.*

$$\begin{aligned} &M \text{ is conflict-free} \\ &\text{iff for all } s \in M \text{ we have } M \cap par(s) \in C_s^{in} \\ &\text{iff } M \subseteq \left\{ s \in S \;\middle|\; M \cap par(s) \in C_s^{in} \right\} \\ &\text{iff } M \subseteq \mathcal{G}'_\Xi(M, M) \\ &\text{iff } (M, M) \leq_t \mathcal{G}_\Xi(M, M) \end{aligned}$$ □

Of course, conflict-free sets $M$ can also be characterised by $M \subseteq G_\Xi(M)$, but we chose the (seemingly more bulky) notation above because we later want to generalise the notion "conflict-free" to three-valued pairs.

### 3.1.2 Model semantics

Much in accordance with logic programming, a model of an ADF is simply a two-valued fixpoint of its associated consequence operator:

**Proposition 3.4.** *For any abstract dialectical framework $\Xi = (S, L, C)$, a set $M \subseteq S$ is a model of $\Xi$ iff $\mathcal{G}_\Xi(M, M) = (M, M)$.*

*Proof.*

$$\begin{aligned} &M \text{ is a model for } \Xi \\ \text{iff } &\text{for each } s \in S \text{ we have } s \in M \text{ iff } M \cap par(s) \in C_s^{in} \\ \text{iff } &M = \left\{ s \in S \;\middle|\; M \cap par(s) \in C_s^{in} \right\} \\ \text{iff } &M = \mathcal{G}'_\Xi(M, M) \\ \text{iff } &\mathcal{G}_\Xi(M, M) = (M, M) \end{aligned}$$

□

Since the correspondence with logic programming is striking, we will use the more specific term "two-valued supported model" from now on.

### 3.1.3 Stable model semantics

Motivated by the same notion of logic programming, Brewka and Woltran [3] defined stable models for bipolar ADFs. When we compare their definition to the general operator-based notion of two-valued stable models, we have to acknowledge a slight mismatch.

**Example 3.1.** Consider the following (bipolar) ADF $\xi = (S, L, C)$ with components $S = \{a, b\}$, $L = \{(a, a), (a, b), (b, b)\}$ and $C_a^{in} = \{\{a\}\}$ and $C_b^{in} = \{\emptyset, \{a\}, \{b\}\}$. In words, $a$ supports itself while $a$ and $b$ jointly attack $b$. The set $M = \{b\}$ is a BW-stable model of $\xi$: The reduct $\xi^M$ is given by the triple $(\{b\}, \emptyset, \{C'_b\})$ with $C'_b = \{\emptyset\}$, an ADF where $b$ is always *in*. (The link $(b, b)$ is not in the reduct because it is attacking in $\xi$.) However, the operator $\mathcal{G}_\xi$ does not have a two-valued stable model: when trying to reconstruct the upper bound $\{b\}$, we get $\mathcal{G}'_\xi(\emptyset, \{b\}) = \emptyset$ since $b$ attacks itself and thus its containment in the upper bound prevents its inclusion in the new lower bound, as witnessed by $par(b) \cap \{b\} = \{b\} \neq \emptyset$. (Interestingly, this example also shows that M-stable models are not necessarily M-supported: $\xi$ has the single M-stable model $(\emptyset, \{b\})$ and the two M-supported models $(\{a\}, \{a, b\})$ and $(\{b\}, \{b\})$.)

So while there are ADFs with BW-stable models which are not two-valued stable models of the ADF's approximating operator, we can establish an inclusion relation for the converse direction: any operator-based two-valued stable model of an ADF is also a BW-stable model of the ADF. To show this, we first need a lemma that relates the operators $\mathcal{G}'_{\Xi^M}$ and $\mathcal{G}'_\Xi(\cdot, M)$ whenever $M$ is a model of $\Xi$.

**Lemma 3.5.** *Let $\Xi = (S, L, C)$ be a bipolar ADF and $(M, M)$ be a two-valued supported model for $\Xi$. For any $X \subseteq S$,*

$$\mathcal{G}'_\Xi(X, M) = X \quad \textit{implies} \quad \mathcal{G}'_{\Xi^M}(X, X) = (X, X)$$

*Proof.* Recall that the reduct of $\Xi$ with $M$ is defined by $\Xi^M = (S^M, L^M, C^M)$ with $S^M = S \cap M$, reduced links $L^M = \left\{(r,s) \mid r, s \in S^M, (r,s) \in L^+\right\}$ and for each $s \in S^M$ and $B \subseteq S^M$, we have $C_s^M(B) = \mathit{in}$ iff $C_s(B) = \mathit{in}$. Now for each $s \in S$ denote by $P_s$ the parent nodes of $s$ with respect to $L$ and for $s \in M$ by $P_s^M$ the parent nodes of $s$ with respect to $L^M$. (Note that all links in $L^M$ are supporting and therefore all statements in $P_s^M$ support $s$.) Let $X \subseteq S$.

$$\begin{aligned} X = \mathcal{G}'_\Xi(X,M) \text{ iff } & X = \left\{s \in S \mid P_s^M \in C_s, P_s^M \subseteq X, (P_s \setminus P_s^M) \cap M = \emptyset\right\} \\ \text{implies } & X = \left\{s \in M \mid P_s^M \in C_s^M, P_s^M \subseteq X, (P_s^M \setminus P_s^M) \cap M = \emptyset\right\} \\ \text{iff } & X = \left\{s \in M \mid P_s^M \subseteq X\right\} \\ \text{iff } & X = \mathcal{G}'_{\Xi^M}(X,X) \\ \text{iff } & (X,X) = \mathcal{G}_{\Xi^M}(X,X) \end{aligned}$$ □

Using the lemma, it is easy to show that BW-stable models subsume operator-based two-valued stable models.

**Proposition 3.6.** *For any bipolar abstract dialectical framework $\Xi = (S,L,C)$, a set $M \subseteq S$ is a BW-stable model of $\Xi$ whenever $\mathcal{SG}_\Xi(M,M) = (M,M)$.*

*Proof.* By Lemma 3.5 above, we know that $(X,X)$ is the least exact fixpoint of $\mathcal{G}_{\Xi^M}$ whenever $X$ is the least fixpoint of $\mathcal{G}'_\Xi(\cdot,M)$. The equivalences below follow.

$$\begin{aligned} & M \text{ is a BW-stable model of } \Xi \\ \text{iff } & M \text{ is a model of } \Xi \text{ and the least model of } \Xi^M \\ \text{iff } & \mathcal{G}_\Xi(M,M) = (M,M) \text{ is the least exact fixpoint of } \mathcal{G}_{\Xi^M} \\ \text{if } & M \text{ is the least fixpoint of } \mathcal{G}'_\Xi(\cdot,M) \\ \text{iff } & M = \mathit{lfp}(\mathcal{G}'_\Xi(\cdot,M)) = c\mathcal{G}_\Xi(M) \\ \text{iff } & \mathcal{SG}_\Xi(M,M) = (M,M) \end{aligned}$$ □

The mismatch noticed in Example 3.1 does not depend on our definition of the four-valued approximating operator: the ADF presented there also does not allow for *ultimate* two-valued stable models, although the model notion of Brewka and Woltran [3] is perfectly captured by the two-valued one-step ADF consequence operator, which also gives rise to ADF's ultimate family of semantics. Put another way, if we take the model notion from Brewka and Woltran [3] and apply to it the transformations of Denecker et al. [8], we arrive at an ultimate stable model semantics which is demonstrably different from BW-stable models.

Thus at the current point, we have two different stable model semantics at our disposal – operator-based two-valued stable models and BW-stable models. The following example shows that the BW-stable semantics admits too many models, since there are ADFs which admit for BW-stable models where one is a proper subset of another.

**Example 3.2.** Consider the following (bipolar) ADF $\xi = (S,L,C)$ with components $S = \{a,b\}$, $L = \{(a,b),(b,b)\}$ and $C_a^{in} = \{\emptyset\}$ and $C_b^{in} = \{\emptyset, \{b\}, \{a,b\}\}$. In words, $a$ is always *in* and attacks $b$, which however can support itself. The ADF $\xi$ has two BW-stable models, $M_1 = \{a\}$ and $M_2 = \{a,b\}$: The reduct of $\xi$ with $M_1$ is given by $\xi^{M_1} = (\{a\}, \emptyset, C^{M_1})$ with $C_a^{M_1} = \{\emptyset\}$, thus its least model is $\{a\} = M_1$. For the second stable model $M_2 = \{a,b\}$, the reduct of $\xi$ with $M_2$ is given by $\xi^{M_2} = (S, \{(b,b)\}, C^{M_2})$ with $C_a^{M_2} = \{\emptyset\}$ and $C_b^{M_2} = \{\emptyset, \{b\}\}$. (Note that the link $(b,b)$ is both supporting and attacking, thus the ADF is not bipolar.) It is easy to see that $\{a,b\} = M_2$ is the least model of this ADF. In contrast, the approximating operator $\mathcal{G}_\xi$ associated with $\xi$ admits only the single two-valued stable model $(\{a\},\{a\})$.

The problem with this example is that it marks a departure from the usual nomenclature of nonmonotonic reasoning formalisms: in logic programming, two distinct stable models cannot be in a subset-relationship; likewise in Reiter's default logic, two distinct extensions of a default theory cannot be in a subset-relationship. With our operator-based definition of two-valued stable models for ADFs, this property comes for free:

**Proposition 3.7.** *Let $(L, \sqsubseteq)$ be a complete lattice and $\mathcal{O}$ an approximating operator on the bilattice $(L^2, \leq_i)$. For any $x, y \in L$ with $\mathcal{SO}(x, x) = (x, x)$ and $\mathcal{SO}(y, y) = (y, y)$, we have that $x \sqsubseteq y$ implies $x = y$.*

*Proof.* Let $x, y \in L$ with $\mathcal{SO}(x, x) = (x, x)$, $\mathcal{SO}(y, y) = (y, y)$ and $x \sqsubseteq y$. Since $\mathcal{O}$ is antimonotone in the second component, we have $\mathcal{O}(x, y) \sqsubseteq \mathcal{O}(x, x) = x$ and $x$ is a prefixpoint of $\mathcal{O}(\cdot, y)$. Now $y$ is the least prefixpoint of $\mathcal{O}(\cdot, y)$ and thus $y \sqsubseteq x$. □

Together with Example 3.2, this result means that there is no approximating operator for which Definition 2.1 can reconstruct BW-stable models.

Finally, our operator-based definition of two-valued stable models easily gives rise to an equivalent reduct-based definition of the same concept: in operator terms, $M$ is a two-valued stable model of $\mathcal{G}_\Xi$ iff $M$ is the least fixpoint of the operator $\mathcal{G}'_\Xi(\cdot, M)$. To define a reduct, we have to find the ADF associated to this consequence operator defined for $X \subseteq S$ by

$$\mathcal{G}'_\Xi(X, M) = \left\{ s \in S \;\middle|\; B \in C_s^{in}, B \subseteq X, (par(s) \setminus B) \cap M = \emptyset \right\}$$

Our new operator-inspired reduct now just has to mimic the way the operator enforces the upper bound $M$. This is achieved by the definition below, which notably works for all ADFs, bipolar or not.

**Definition 3.2.** Let $\Xi = (S, L, C^{in})$ be an abstract dialectical framework. A set $M \subseteq S$ is a *stable model* of $\Xi$ iff it is the unique least model of the reduced ADF $\Xi_M = (S, L, C_M^{in})$ with

$$B \in C_{M,s}^{in} \text{ iff } B \in C_s^{in}, (par(s) \setminus B) \cap M = \emptyset$$

Intuitively, the reduct only changes the acceptance functions of statements such that accepting parent configurations that rely on some statement from $M$ being *out* are discarded (since the statements in $M$ are by virtue *in*). If the reduced ADF has a unique least model, and this least model coincides with $M$, then $M$ is a stable model of the original ADF. It is easy to show that this new reduct-based definition of a stable model coincides with our operator-based definition of two-valued stable models.

**Proposition 3.8.** *Let $\Xi = (S, L, C^{in})$ be an abstract dialectical framework and $M \subseteq S$. $(M, M)$ is a two-valued stable model of $\mathcal{G}_\Xi$ iff $M$ is a stable model of $\Xi$.*

*Proof.* First observe that we find the two-valued consequence operator of the reduct $\Xi_M$ given for any $X \subseteq S$ by

$$\begin{aligned} G_{\Xi_M}(X) = \{ s \in S \mid\ & B \in C_s^{in}, (par(s) \setminus B) \cap M = \emptyset, \\ & B \subseteq X, (par(s) \setminus B) \cap X = \emptyset \} \end{aligned}$$

Hence $X \subseteq M$ implies $G_{\Xi_M}(X) = \mathcal{G}'_\Xi(X, M)$ and the two operators $G_{\Xi_M}$ and $\mathcal{G}'_\Xi(\cdot, M)$ coincide for all subsets of $M$. In particular, $M$ is the least fixpoint of $\mathcal{G}'_\Xi(\cdot, M)$ iff $M$ is the least fixpoint of $G_{\Xi_M}$. (The least fixpoint of $\mathcal{G}'_{\Xi}(\cdot, M)$ always exists since the operator is monotone in $(2^S, \subseteq)$.) Now

$$\begin{aligned}
&(M, M) \text{ is a two-valued stable model of } \mathcal{G}_{\Xi} \\
\text{iff } &M \text{ is the least fixpoint of } \mathcal{G}'_{\Xi}(\cdot, M) \\
\text{iff } &M \text{ is the least fixpoint of } G_{\Xi_M} \\
\text{iff } &M \text{ is the least model of } \Xi_M \\
\text{iff } &M \text{ is a stable model of } \Xi
\end{aligned}$$

□

**Example 3.3.** Let us reconsider the problematic ADF from Example 3.2, $\xi = (S, L, C)$ with components $S = \{a, b\}$, $L = \{(a, b), (b, b)\}$ and $C_a^{in} = \{\emptyset\}$ and $C_b^{in} = \{\emptyset, \{b\}, \{a, b\}\}$.

The (new) reduct of $\xi$ with $M_1 = \{a, b\}$ is given by $\xi_{M_1} = (S, L, C_{M_1})$ with $C_{M_1,a}^{in} = \{\emptyset\}$ and $C_{M_1,b}^{in} = \{\{a, b\}\}$. It is easy to see that $\{a\} \neq M_1$ is the least model of this ADF and $M_1$ is not a stable model of $\xi$.

The (new) reduct of $\xi$ with $M_2 = \{a\}$ is given by $\xi_{M_2} = (S, L, C_{M_2}^{in})$ with $C_{M_2,a}^{in} = \{\emptyset\}$ and $C_{M_2,b}^{in} = \{\{a, b\}\}$. Its least model is $\{a\} = M_2$ and $M_2$ is thus a stable model of $\xi$, just as expected.

### 3.1.4 Admissible sets

For the generalisation of admissibility provided by Brewka and Woltran [3], the picture is not quite as clear. Firstly, for the special case of Dung argumentation frameworks, any stable extension of an AF is admissible. So we should naturally expect that all ADF generalisations of stable AF extensions are also (the ADF generalisation of) admissible; more specifically, since for AF-based ADFs we have that stable extensions coincide with two-valued supported models of the ADF, for an ADF generalisation of admissibility we should expect that all two-valued supported models of the ADF are also admissible. But this is not the case for the generalisation of admissibility of Brewka and Woltran [3]. Recall that a set $M$ is BW-admissible iff there exists an $R \subseteq S$ such that $M$ is a stable model of $\Xi - R$.

**Example 3.4.** Consider the simplest ADF with a self-supporting cycle between two arguments, $\xi = (S, L, C)$ with $S = \{a, b\}$, $L = \{(a, b), (b, a)\}$ and $C_a^{in} = \{\{b\}\}$, $C_b^{in} = \{\{a\}\}$. In other words, the links between $a$ and $b$ are both supporting. Hence the set $\{a, b\}$ is a (two-valued supported) model of $\xi$, but it is not BW-admissible: $\{a, b\}$ is not a stable model of $\xi$ or any subframework of $\xi$.

It might seem that BW-admissibility is just too restrictive and could be fixed by weakening the definition. One possibility may be to replace "stable" in the definition of BW-admissibility by "supported." But, as the following example shows, already the current, stable-based definition of BW-admissibility considers too many sets to be admissible.

**Example 3.5.** Consider the (bipolar) ADF $\xi = (S, L, C)$ with statements $S = \{a, b, c, d\}$, links $L = \{(b, a), (c, a), (d, c)\}$ and acceptance conditions $C_a^{in} = \{\emptyset, \{b\}, \{c\}\}$, $C_b^{in} = \{\emptyset\}$, $C_c^{in} = \{\{d\}\}$ and $C_d^{in} = \{\emptyset\}$. In words, there is a joint attack of $b$ and $c$ on $a$ – $a$ is *out* if both $b$ and $c$ are *in*, and $a$ is *in* otherwise. Statements $b$ and $d$ are always *in*, and $c$ is *in* if $d$ is. This ADF $\xi$ has the BW-admissible set $M = \{a, b\}$: Taking $R = \{d\}$, we see that there are no attacks from $R$ to $M$. Furthermore, the ADF $\xi - R = \xi' = (S', L', C')$ is given by $S' = \{a, b, c\}$, $L' = \{(b, a), (c, a)\}$ and $C'_a = \{\emptyset, \{b\}, \{c\}\}$, $C'_b = \{\emptyset\}$ and $C'_c = \{\}$. This ADF $\xi'$ has the stable model $\{a, b\}$, which is easily verified when looking at the reduct $\xi'^M = (\{a, b\}, \emptyset, C'^M)$ where $C_a'^M = C_b'^M = \{\emptyset\}$. So in a sense, the set $\{a, b\}$ being admissible depends on the removal of $\{d\}$, in which case the only support of $c$ is removed and the joint attack on $a$ cannot happen. But $d$ is by definition of its acceptance condition always *in*, so no reasonable semantics could ever label it *out*, and consequently the condition upon which BW-admissibility of $\{a, b\}$ hinges can never become true.[4]

There is an alternative characterisation of admissibility which satisfies all of our abovementioned criteria. That is, all two-valued supported models of an ADF are admissible in our new sense; and for the ADF from Example 3.5, the undesired BW-admissible set from above is not admissible according to this new definition. As a much more important property, it is defined for *all* ADFs and not only bipolar ones. It is also a generalisation of AF admissibility, as will be shown in Section 4. The following definition of admissible requires the pair in question to be consistent, three-valued conflict-free and the $\Xi$-revision of the pair should be at least as precise as the pair itself.

**Definition 3.3.** For any ADF $\Xi = (S, L, C)$, a consistent pair $(M, N)$ is *admissible in* $\Xi$ iff $(M, N) \leq_t \mathcal{G}_\Xi(M, N)$ and $(M, N) \leq_i \mathcal{G}_\Xi(M, N)$.

It is clear that all two-valued admissible pairs are conflict-free. Since for any two-valued supported model $M$ we have $(M, M) = \mathcal{G}_\Xi(M, M)$ it is also immediate that all two-valued supported models of an ADF are (three-valued supported models and in turn) admissible pairs.

**Example 2.1** (Continued)**.** Our running example ADF $D$ has the following admissible pairs:

$$\begin{array}{llll}
(\emptyset, \{a, b, c, d\}) & (\{a\}, \{a, b, c, d\}) & (\{b\}, \{a, b, c\}) & (\{a, d\}, \{a, d\}) \\
(\emptyset, \{a, d\}) & (\{a\}, \{a, d\}) & (\{a, b\}, \{a, b, c\}) & (\{a, b, c\}, \{a, b, c\})
\end{array}$$

These pairs include all three-valued supported models, in particular all two-valued supported models and also, as we will see later, the Kripke-Kleene semantics of $D$.

### 3.1.5 Preferred semantics

In principle, there could be different ways to define the preferred semantics for ADFs: (1) the argumentation way of taking all model candidates that are maximally admissible; (2) the logic-programming way of maximising over three-valued supported models. It is clear that any preferred pair derived according to (2) is also preferred in the sense of (1) since any three-valued supported model is admissible. But – as we will show next – the converse also holds, so it is inessential which of these two definitions we pick. This even holds for any approximating operator on a complete lattice, as is shown by the theorem below; in AF-speak, it expresses the operator generalisation of "all preferred extensions are complete."

**Theorem 3.9.** *Let* $(L, \sqsubseteq)$ *be a complete lattice and* $\mathcal{O}$ *be an approximating operator on* $(L^2, \leq_i)$*. Any* $\leq_i$*-maximal admissible pair for* $\mathcal{O}$ *is a three-valued supported model for* $\mathcal{O}$*.*

*Proof.* Let $(x, y)$ be an $\leq_i$-maximal admissible pair, that is, $(x, y) \leq_t \mathcal{O}(x, y)$ as well as $(x, y) \leq_i \mathcal{O}(x, y)$ and there is no admissible pair $(\hat{x}, \hat{y})$ with $(x, y) <_i (\hat{x}, \hat{y})$. By admissibility of $(x, y)$, we get $x \sqsubseteq \mathcal{O}'(x, y)$ and $\mathcal{O}'(y, x) = y$. By $x \sqsubseteq y$ and $\mathcal{O}'$ being monotone in the first and antimonotone in the second component, we obtain the following picture.

$$\begin{tikzcd}
 & & \mathcal{O}'(x,x) \arrow[dr, phantom, "\sqsubseteq" sloped] & & \\
x \arrow[r, phantom, "\sqsubseteq"] & \mathcal{O}'(x,y) \arrow[ur, phantom, "\sqsubseteq" sloped] \arrow[dr, phantom, "\sqsubseteq" sloped] & & \mathcal{O}'(y,x) \arrow[r, phantom, "="] & y \\
 & & \mathcal{O}'(y,y) \arrow[ur, phantom, "\sqsubseteq" sloped] & &
\end{tikzcd}$$

[4] Incidentally, $\{a, b\}$ is also a BW-preferred model which does not contain the BW-well-founded model $\{b, c, d\}$. Since the grounded AF extension is always contained in any preferred AF extension, Example 3.5 also hints at another unexpected (non-)relation between the generalised ADF semantics of Brewka and Woltran [3].

Assume to the contrary of what we have to show that $(x, y) \neq \mathcal{O}(x, y)$. Since $x \sqsubseteq \mathcal{O}'(x, y)$ and $\mathcal{O}'(y, x) = y$, this means $\mathcal{O}'(x, y) \not\sqsubseteq x$. We construct an admissible pair $(\hat{x}, \hat{y})$ with $(x, y) <_i (\hat{x}, \hat{y})$. First, set $\hat{x} = x \sqcup \mathcal{O}'(x, y)$. Now $\mathcal{O}'(x, y) \sqsubseteq y$ and thus $\hat{x} \sqsubseteq y$. This means that $(\{z \mid \hat{x} \sqsubseteq z \sqsubseteq y\}, \sqsubseteq)$ is a complete lattice. We will show next that $\mathcal{O}'(\cdot, \hat{x})$ is an operator on this lattice: let $\hat{x} \sqsubseteq z \sqsubseteq y$, we show $\hat{x} \sqsubseteq \mathcal{O}'(z, \hat{x}) \sqsubseteq y$. As a start, $x \sqsubseteq \mathcal{O}'(x, y)$ and the definition of $\hat{x}$ yields $\hat{x} \sqsubseteq \mathcal{O}'(x, y)$. Now $x \sqsubseteq \hat{x} \sqsubseteq y$ means $(x, y) \leq_i (\hat{x}, \hat{x})$ and thus $\mathcal{O}'(x, y) \sqsubseteq \mathcal{O}'(\hat{x}, \hat{x})$. Next, from $\hat{x} \sqsubseteq z \sqsubseteq y$ it follows that $\mathcal{O}'(\hat{x}, \hat{x}) \sqsubseteq \mathcal{O}'(z, \hat{x}) \sqsubseteq \mathcal{O}'(y, \hat{x})$. Finally, $x \sqsubseteq \hat{x}$ and antimonotonicity yield $\mathcal{O}'(y, \hat{x}) \sqsubseteq \mathcal{O}'(y, x) = y$. In combination, we get: $\hat{x} \sqsubseteq \mathcal{O}'(x, y) \sqsubseteq \mathcal{O}'(\hat{x}, \hat{x}) \sqsubseteq \mathcal{O}'(z, \hat{x}) \sqsubseteq \mathcal{O}'(y, \hat{x}) \sqsubseteq y$. Hence $\mathcal{O}'(\cdot, \hat{x})$ is a $\sqsubseteq$-monotone operator on the lattice $(\{z \mid \hat{x} \sqsubseteq z \sqsubseteq y\}, \sqsubseteq)$ and thus has a least fixpoint. Define $\hat{y} \overset{\text{def}}{=} \mathit{lfp}(\mathcal{O}'(\cdot, \hat{x}))$. We now show that $(\hat{x}, \hat{y})$ is admissible. First, $\hat{x} \sqsubseteq \hat{y} \sqsubseteq y$ is immediate from the lattice in which $\hat{y}$ is defined. By monotonicity of $\mathcal{O}'$ we get $\hat{x} \sqsubseteq \mathcal{O}'(x, y) \sqsubseteq \mathcal{O}'(\hat{x}, y) \sqsubseteq \mathcal{O}'(\hat{x}, \hat{y})$. Finally, $\mathcal{O}'(\hat{y}, \hat{x}) = \hat{y}$ by definition. Hence $(\hat{x}, \hat{y})$ is admissible and by $x \sqsubseteq \hat{x}$, $\hat{x} \sqsubseteq \mathcal{O}'(x, y) \not\sqsubseteq x$ and $\hat{y} \sqsubseteq y$ we have $(x, y) <_i (\hat{x}, \hat{y})$. Contradiction. □

As an immediate consequence, we have the result that all maximal admissible ADF models are three-valued supported (as we will see, "complete") models.

**Corollary 3.10.** *Let $\Xi$ be an abstract dialectical framework. Any $\leq_i$-maximal admissible pair is a three-valued supported model.*

This leads to the generalisation of AF preferred semantics for abstract dialectical frameworks (including non-bipolar ones): they are simply M-supported models of $\mathcal{G}_\Xi$, that is, $\leq_i$-maximal fixpoints of $\mathcal{G}_\Xi$. Since supported and stable semantics coincide for argumentation frameworks, another suitable candidate for generalising preferred semantics is the M-stable semantics for ADFs, that is, $\leq_i$-maximal fixpoints of $\mathcal{SG}_\Xi$.

**Well-founded semantics** In order to generalise the grounded semantics from AFs to ADFs, Brewka and Woltran [3] introduced – for an ADF $\Xi = (S, L, C)$ – the operator $\Gamma_\Xi$ on the bilattice $(2^S \times 2^S, \leq_i)$. Motivated by naming conventions from logic programming, they decided to call (the lower bound of) the least fixpoint of $\Gamma_\Xi$ the "well-founded model" of an ADF. As our next result shows, their intuition of defining the operator was on the spot – they defined the most precise approximation of the two-valued ADF immediate consequence operator $G_\Xi$.

**Lemma 3.11.** *For any ADF $\Xi$, the operator $\Gamma_\Xi$ is the ultimate approximation of $G_\Xi$.*

*Proof.* Recall that for $\Xi = (S, L, C)$ the operator $\Gamma_\Xi : 2^S \times 2^S \to 2^S \times 2^S$ is given by $(X, Y) \mapsto (\Gamma'_\Xi(X, Y), \Gamma''_\Xi(X, Y))$, where

$$\Gamma'_\Xi(X, Y) = \left\{ s \in S \;\middle|\; \text{for all } X \subseteq Z \subseteq Y, \text{we have } Z \cap \mathit{par}(s) \in C_s^{in} \right\}$$
$$\Gamma''_\Xi(X, Y) = \left\{ s \in S \;\middle|\; \text{there exists } X \subseteq Z \subseteq Y \text{ with } Z \cap \mathit{par}(s) \in C_s^{in} \right\}$$

Now by [8, Theorem 5.6], for $X \subseteq Y \subseteq S$, the ultimate approximation $\mathcal{U}_\Xi$ of the operator $G_\Xi$ is characterised by $\mathcal{U}_\Xi(X, Y) = (\mathcal{U}'_\Xi(X, Y), \mathcal{U}''_\Xi(X, Y))$ with

$$\mathcal{U}'_\Xi(X, Y) = \bigcap \{G_\Xi(Z) \mid X \subseteq Z \subseteq Y\}$$
$$\mathcal{U}''_\Xi(X, Y) = \bigcup \{G_\Xi(Z) \mid X \subseteq Z \subseteq Y\}$$

By Lemma 3.1, we know that for any $s \in S$ and $Z \subseteq S$ we find $Z \cap par(s) \in C_s^{in}$ iff $s \in \mathcal{G}_\Xi(Z)$, which leads to the equalities

$$\begin{aligned}
\mathcal{U}_\Xi'(X,Y) &= \bigcap \{\mathcal{G}_\Xi(Z) \mid X \subseteq Z \subseteq Y\} \\
&= \{s \in S \mid \text{for all } X \subseteq Z \subseteq Y, \text{we have } s \in \mathcal{G}_\Xi(Z)\} \\
&= \left\{s \in S \;\middle|\; \text{for all } X \subseteq Z \subseteq Y, \text{we have } Z \cap par(s) \in C_s^{in}\right\} \\
&= \Gamma_\Xi'(X,Y)
\end{aligned}$$

and, likewise for the upper bound,

$$\begin{aligned}
\mathcal{U}_\Xi''(X,Y) &= \bigcup \{\mathcal{G}_\Xi(Z) \mid X \subseteq Z \subseteq Y\} \\
&= \{s \in S \mid \text{there exists } X \subseteq Z \subseteq Y \text{ with } s \in \mathcal{G}_\Xi(Z)\} \\
&= \left\{s \in S \;\middle|\; \text{there exists } X \subseteq Z \subseteq Y \text{ with } Z \cap par(s) \in C_s^{in}\right\} \\
&= \Gamma_\Xi''(X,Y)
\end{aligned}$$

which proves the claim. □

This lemma immediately entails that what Brewka and Woltran [3] called "well-founded" is what DMT call the ultimate Kripke-Kleene semantics.

**Corollary 3.12.** *For any ADF $\Xi$, its BW-well-founded semantics coincides with its ultimate Kripke-Kleene semantics.*

The well-founded semantics of $\Xi$ in the usual sense (the least fixpoint of the stable operator $\mathcal{SG}_\Xi$) hence may differ from the BW-well-founded semantics.

**Example 2.1** (Continued)**.** Recall that the ultimate Kripke-Kleene semantics of $D$ is given by the pair $(\{a\}, \{a,b,c,d\})$ (the least model of the operator $\mathcal{U}_D = \Gamma_D$). The well-founded semantics of $D$ in the logic-programming sense is given by the pair $(\{a,d\}, \{a,d\})$. Since this pair is exact, it also represents the unique two-valued stable model of $D$.

We have seen how the characteristic operator of an ADF can be used to redefine several existing ADF semantics. The remaining operator-based semantics that we did not yet talk about therefore present new semantics for ADFs. Among them, we generalised complete AF extensions to ADFs (three-valued supported/stable models) which will be explored in more detail in the AF section.

## 3.2 From ADFs to Logic Programs

We now use the four-valued one-step ADF consequence operator to determine the relationship between ADFs and logic programs. As it turns out, there is a straightforward polynomial and modular translation from ADFs to logic programs which is additionally faithful with respect to all operator-based semantics. The translation creates a logic program rule for each statement of a given ADF $\Xi$. The rule body for statement $s$ is satisfied whenever for some $M \subseteq par(s)$, the statements in $M$ are *in* and the remaining parents are *out*.

**Definition 3.4.** Let $\Xi = (S, L, C^{in})$ be an ADF. Define its *standard logic program* as follows.

$$\Pi(\Xi) \stackrel{\text{def}}{=} \left\{s \leftarrow (M \cup not\,(par(s) \setminus M)) \;\middle|\; s \in S, M \in C_s^{in}\right\}$$

**Example 2.1** (Continued)**.** The standard logic program $\Pi(D)$ of our running example ADF $D$ is given by

$$a \leftarrow \emptyset \qquad b \leftarrow b \qquad c \leftarrow \{a, b\} \qquad d \leftarrow \mathit{not}\, b$$

As another illustrative example, we look at the ADF where we observed a mismatch between BW-stable models and operator-based two-valued stable models.

**Example 3.6.** The ADF $\xi$ from Example 3.1 is translated into the logic program consisting of the following rules:

$$a \leftarrow a \qquad b \leftarrow \{\mathit{not}\, a, \mathit{not}\, b\} \qquad b \leftarrow \{a, \mathit{not}\, b\} \qquad b \leftarrow \{b, \mathit{not}\, a\}$$

This somewhat obviates why there is no two-valued stable model for $\xi$: the only candidate for deriving $b$ in some reduct program is the last rule, which however circularly requires $b$ itself.

The next lemma shows that the term "standard logic program" is well-chosen, since the translation is faithful with respect to all operator-based semantics: the associated approximating operators of an ADF and its standard logic program are identical. The term $\overline{B}$ below denotes the complement of $B$ with respect to the parents of $s$, that is, $\overline{B} = \mathit{par}(s) \setminus B$.

**Lemma 3.13.** *For any ADF $\Xi = (S, L, C^{in})$, we find that $\mathcal{G}_\Xi = \mathcal{T}_{\Pi(\Xi)}$.*

*Proof.* Let $X, Y \subseteq S$. We show $\mathcal{G}'_\Xi(X, Y) = \mathcal{T}'_{\Pi(\Xi)}(X, Y)$.

$$\begin{aligned} \mathcal{G}'_\Xi(X, Y) &= \left\{ s \in S \;\middle|\; B \in C^{in}_s, B \subseteq X, \overline{B} \cap Y = \emptyset \right\} \\ &= \left\{ s \in S \;\middle|\; s \leftarrow (B \cup \mathit{not}\, \overline{B}) \in \Pi(\Xi), B \subseteq X, \overline{B} \cap Y = \emptyset \right\} \\ &= \left\{ s \in S \;\middle|\; s \leftarrow M \in \Pi(\Xi), B = M^+ \subseteq X, \overline{B} = M^-, M^- \cap Y = \emptyset \right\} \\ &= \mathcal{T}'_{\Pi(\Xi)}(X, Y) \end{aligned}$$ □

This result yields immediate correspondence of all operator-based semantics of an ADF $\Xi$ with the respective semantics of its standard logic program $\Pi(\Xi)$.

**Theorem 3.14.** *Let $\Xi = (S, L, C^{in})$ be an abstract dialectical framework and $\Pi(\Xi)$ its standard logic program. Then $\Xi$ and $\Pi(\Xi)$ coincide on all semantics based on their approximation operators.*

In particular, $G_\Xi = T_{\Pi(\Xi)}$ and an ADF and its standard logic program also agree on all semantics derived from the ultimate approximation of their two-valued operators. These results obviate that propositional normal logic programs are at least as expressive as abstract dialectical frameworks in a very strong sense: there exists a *single* translation that preserves models in a whole *type* of semantics. Furthermore, the translation can be computed in polynomial time and is modular with respect to statements.

More precisely, let $\Xi_1 = (S_1, L_1, C^{in}_1)$ and $\Xi_2 = (S_2, L_2, C^{in}_2)$ be ADFs such that $S_1 \cap S_2 = \emptyset$. Then the union of the two ADFs is defined as $\Xi_1 \cup \Xi_2 \overset{\text{def}}{=} (S_1 \cup S_2, L_1 \cup L_2, C^{in}_1 \cup C^{in}_2)$. For such pairs of ADFs we indeed observe that the translation is modular: $\Pi(\Xi_1 \cup \Xi_2) = \Pi(\Xi_1) \cup \Pi(\Xi_2)$.

But it is not straightforward to define the union of two ADFs when they share statements:

**Example 3.7.** Consider the ADFs $\xi_1 = (S_1, L_1, C^{in}_1)$ with $S_1 = \{a, b\}$, $L_1 = \{(b, a)\}$, $C^{in}_{1,a} = \{\{b\}\}$ and $C^{in}_{1,b} = \{\emptyset\}$ (in words, $b$ is always *in* and supports $a$); and $\xi_2 = (S_2, L_2, C^{in}_2)$ with $S_2 = \{a, c\}$, $L_2 = \{(c, a)\}$, $C^{in}_{2,a} = \{\{c\}\}$ and $C^{in}_{2,c} = \{\emptyset\}$ (in words, $c$ is always *in* and supports $a$). In both frameworks, the common statement $a$ is supported by a statement which is always *in*. Consequently, $a$ is always *in* for every model of every semantics in both ADFs. However, the union of the acceptance functions' characteristic sets is $C^{in}_{1,a} \cup C^{in}_{2,a} = \{\{b\}, \{c\}\}$, and thus in the union ADF, statement $a$ is always *out* since both parents are always *in*. The undesired result in this case is that $a$ is always accepted in the two constituent ADFs but not accepted in their union, although this union should be expected to exhibit some kind of disjunctive acceptance with respect to its constituents. (For comparison, note that $\Pi(\xi_1) = \{a \leftarrow b, b \leftarrow \emptyset\}$ and $\Pi(\xi_2) = \{a \leftarrow c, c \leftarrow \emptyset\}$, whence $a$ is contained in the single (two-valued) stable model $\{a, b, c\}$ of $\Pi(\xi_1) \cup \Pi(\xi_2)$.)

Of course, the example above would work if we represented acceptance conditions by formulas $\varphi_{1,a} = b$ and $\varphi_{2,a} = c$: then in the union of the two ADFs the acceptance formula is given by the disjunction $\varphi_{1,a} \vee \varphi_{2,a} = b \vee c$ which has the desired set of models $\{\{b\}, \{c\}, \{b, c\}\}$. However, this is dependent on the specific chosen representation of acceptance conditions, namely propositional formulas. For the general case of overlapping sets of statements and an abstract stance with regard to the representation of acceptance conditions, it seems that a more sophisticated procedure for ADF merging is required. This makes it hard to assess a more general type of modularity concerning translations from ADF into logic programs.

## 3.3 From Logic Programs to ADFs

To translate ADFs into logic programs, we essentially had to take the acceptance formulas, transform them into disjunctive normal form and write an LP rule for each disjunct. To translate logic programs into ADFs, this process is reversed: to devise an acceptance function for statement $s$, we take the disjunction of all bodies (read as conjunctions of literals) of rules with head $s$.

**Definition 3.5** (Brewka and Woltran [3])**.** Let $\Pi$ be a normal logic program over a set $A$ of atoms. Define an ADF $\Xi(\Pi) = (A, L, C^{in})$ as follows.

- $L \overset{\text{def}}{=} \{(b, a) \mid a \leftarrow M \in \Pi, b \in M^+ \cup M^-\}$
- For $a \in A$, set $C^{in}_a \overset{\text{def}}{=} \{B \subseteq \mathit{par}(a) \mid a \leftarrow M \in \Pi, M^+ \subseteq B, M^- \cap B = \emptyset\}$.

Alternatively, we could define the acceptance condition of each $a \in A$ by

$$\varphi_a \overset{\text{def}}{=} \bigvee_{a \leftarrow M \in \Pi} \left( \bigwedge_{m \in M^+} m \wedge \bigwedge_{m \in M^-} \neg m \right)$$

Although straightforward, the translation is obviously not modular, since all logic program rules with head $a$ are needed to devise the acceptance condition for statement $a$. Furthermore, the translation is not faithful with respect to three-valued semantics defined by the approximating operator $\mathcal{G}_\Xi$.

**Example 3.8** (Lost in Translation)**.** Consider the following two logic programs over the signature $A = \{a, b, c\}$ that have a common subprogram $\pi = \{c \leftarrow \emptyset, b \leftarrow \mathit{not}\, b\}$:

1. $\pi_1 = \pi \cup \{a \leftarrow b, a \leftarrow c\}$
2. $\pi_2 = \pi \cup \{a \leftarrow \{b, \mathit{not}\, c\},\ a \leftarrow \{c, \mathit{not}\, b\},\ a \leftarrow \{b, c\}\}$

The ADF translations of the two programs are identical: we have $\Xi(\pi_1) = \Xi(\pi_2) = (A, L, C^{in})$ with the obvious links from body atoms to head atoms, $L = \{(b, b), (b, a), (c, a)\}$, statement $b$ being self-attacking, $C^{in}_b = \{\emptyset\}$, statement $c$ being *in* by default, $C^{in}_c = \{\emptyset\}$ and statement $a$ being *in* if $b$ is *in*, $c$ is *in*, or both, $C^{in}_a = \{\{b\}, \{c\}, \{b, c\}\}$. However, the original logic programs

$\pi_1$ and $\pi_2$ do not have the same three-valued models: While the only (three-valued supported) model of $\pi_1$ is $(\{a, c\}, \{a, b, c\})$, the only (three-valued supported) model of $\pi_2$ is $(\{c\}, \{a, b, c\})$. That is, when we force the truth values of $c$ to be true and $b$ to be undefined (in the common subprogram $\pi$), the result is that $a$ is true in $\pi_1$ (the disjunction $b \lor c$ evaluates to true) but undefined in $\pi_2$ (all the conjuncts $b \land \neg c$, $c \land \neg b$ and $b \land c$ evaluate to undefined).

However, the translation is faithful for two-valued supported semantics, as we will show next. Technically, this is proved by establishing a correspondence between the two-valued one-step consequence operators $T_\Pi$ for a logic program $\Pi$ and $G_{\Xi(\Pi)}$ for the logic program's ADF $\Xi(\Pi)$ in the following lemma.

**Lemma 3.15.** *For any normal logic program* $\Pi$*, we have* $T_\Pi = G_{\Xi(\Pi)}$.

*Proof.* Abbreviate $\Xi(\Pi) = \Xi$, let $A$ be the signature of $\Pi$ and let $X, Y \subseteq A$. We show something slightly more general than $G_\Xi(X) = \mathcal{G}'_\Xi(X, X) = \mathcal{T}'_\Pi(X, X) = T_\Pi(X)$.

1. $\mathcal{G}'_\Xi(X, Y) \subseteq \mathcal{T}'_\Pi(X, Y)$: Let $a \in \mathcal{G}'_\Xi(X, Y)$. Then there is a $B \in C_a^{in}$ with $B \subseteq X$ and $\overline{B} \cap Y = \emptyset$. By definition of $\Xi(\Pi)$, there is a $B \subseteq par(a)$ and a rule $a \leftarrow M \in \Pi$ with $M^+ \subseteq B$ and $M^- \cap B = \emptyset$. We have to show that $M^+ \subseteq X$ (this is immediate) and $M^- \cap Y = \emptyset$. Assume to the contrary that there is a $b \in M^- \cap Y$. Then $M^- \cap B = \emptyset$ implies $b \notin B$. Similarly, $\overline{B} \cap Y = \emptyset$ implies that $b \notin \overline{B}$. Thus $b \notin B \cup \overline{B} = par(a)$, which is a contradiction to $b \in M^-$, $a \leftarrow M \in \Pi$ and the definition of $\Xi(\Pi)$.

2. $\mathcal{T}'_\Pi(X, X) \subseteq \mathcal{G}'_\Xi(X, X)$: Let $a \in \mathcal{T}'_\Pi(X, X)$. Then there is a rule $a \leftarrow M \in \Pi$ with $M^+ \subseteq X$ and $M^- \cap X = \emptyset$. Define $B \overset{\text{def}}{=} par(a) \cap X$. We have to show that $B \in C_a^{in}$, $B \subseteq X$ (obvious) and $\overline{B} \cap X = \emptyset$. For the last item, we have that $\overline{B} = par(a) \setminus B = par(a) \setminus (par(a) \cap X) = par(a) \setminus X$, whence $\overline{B} \cap X = \emptyset$. Finally, $a \leftarrow M \in \Pi$ means $M^+ \subseteq par(a)$ and together with $M^+ \subseteq X$ we get $M^+ \subseteq B = par(a) \cap X$. Since $B \subseteq X$, we have $M^- \cap B = \emptyset$. By definition $B \subseteq par(a)$ and thus $B \in C_a^{in}$. □

As an immediate consequence, we get correspondence of two-valued supported models.

**Corollary 3.16.** *Let* $\Pi$ *be a normal logic program over a set* $A$ *of atoms and* $\Xi = \Xi(\Pi)$ *be its associated abstract dialectical framework. For any set* $X \subseteq A$*,* $\mathcal{G}_\Xi(X, X) = (X, X)$ *iff* $\mathcal{T}_\Pi(X, X) = (X, X)$.

As another consequence of the proof of Lemma 3.15, we can also show that LP-based ADFs are sound with respect to two-valued stable models of the LP, that is, any stable model of $\Xi(\Pi)$ is a stable model of $\Pi$.

**Lemma 3.17.** *Let* $\Pi$ *be a normal logic program over a set* $A$ *of atoms and* $\Xi = \Xi(\Pi)$ *be its associated abstract dialectical framework. For any set* $X \subseteq A$*,* $\mathcal{SG}_\Xi(X, X) = (X, X)$ *implies* $\mathcal{ST}_\Pi(X, X) = (X, X)$.

*Proof.* Let $\mathcal{SG}_\Xi(X, X) = (X, X)$. Then $X$ is the least fixpoint of $\mathcal{G}'_\Xi(\cdot, X)$ and in particular $\mathcal{G}'_\Xi(X, X) = X$. Now by Lemma 3.15 above, we get $\mathcal{T}'_\Pi(X, X) = X$ and $X$ is a fixpoint of $\mathcal{T}'_\Pi(\cdot, X)$. It remains to show that $X$ is the least fixpoint of $\mathcal{T}'_\Pi(\cdot, X)$. Let $Y$ be a prefixpoint of $\mathcal{T}'_\Pi(\cdot, X)$, that is, $\mathcal{T}'_\Pi(Y, X) \subseteq Y$. By Item 1 in the proof of Lemma 3.15 we have $\mathcal{G}'_\Xi(Y, X) \subseteq \mathcal{T}'_\Pi(Y, X)$, whence $\mathcal{G}'_\Xi(Y, X) \subseteq Y$ and $Y$ is a prefixpoint of $\mathcal{G}'_\Xi(\cdot, X)$. Since $X$ is the least fixpoint of $\mathcal{G}'_\Xi(\cdot, X)$ and thus also its least prefixpoint, we get $X \subseteq Y$ and thus $X$ is the least (pre)fixpoint of $\mathcal{T}'_\Pi(\cdot, X)$. □

The converse of the lemma does not hold:

**Example 3.9.** Let $\pi = \{a \leftarrow \emptyset, a \leftarrow a\}$. This program has the two-valued stable model $\{a\}$. Its resulting ADF is $\xi = \Xi(\pi) = (\{a\}, \{(a,a)\}, \{C_a^{in}\})$ with $C_a^{in} = \{\emptyset, \{a\}\}$. Interestingly, the link $(a,a)$ is both supporting and attacking. When trying to reconstruct the (LP) stable model $\{a\}$, we observe that $\mathcal{G}'_\xi(\emptyset, \{a\}) = \emptyset$ and $\{a\}$ is not a (ADF) stable model for $\xi$.

As much more interesting consequence of Lemma 3.15, it follows that the ultimate approximations of $\mathcal{T}_\Pi$ and $G_{\Xi(\Pi)}$ are identical, thus $\Pi$ and $\Xi(\Pi)$ also coincide on all ultimate semantics, including ultimate stable models. This means that whatever "goes missing" in the translation from $\Pi$ to $\Xi(\Pi)$ can be recovered by the construction of the ultimate approximation. This should however be taken with a grain of salt, since the ultimate versions of approximation semantics are generally accompanied by higher computational costs [8]. So while information thrown away through translation can be recovered, it seems much more economic to keep the information during translation instead of paying for a subsequent reconstruction.

# 4 A Special Case: Argumentation Frameworks

In this section we look at the subset of ADFs which corresponds to AFs. Recall that for AFs, the original lattice of interest $(2^A, \subseteq)$ considers sets of arguments and the subset relation. The corresponding bilattice $(2^A \times 2^A, \leq_i)$ is concerned with pairs of sets of arguments and ordered by the information ordering. The elements of this bilattice generalise three-valued labellings [5] to the four-valued case: for a pair $(S, P)$, the arguments in $S \cap P$ are *in*, those in $\overline{S \cup P}$ are *out*, those in $P \setminus S$ are *undecided* and those in $S \setminus P$ get the new label *inconsistent*. Consistent pairs (those $(S, P)$ with $S \subseteq P$) obviously are three-valued labellings, where exactly all arguments in $S$ are *in*.

As our first observation, we note that the approximating operator that Definition 3.1 assigns to the ADF of an AF $\Theta$ is also a special case of an operator: it is the canonical approximation of $U_\Theta$, the operator assigning to a set $S$ of arguments all the arguments from $A$ which are not attacked by $S$.

**Proposition 4.1.** *For any argumentation framework $\Theta = (A, R)$ and sets $X, Y \subseteq A$, we have $\mathcal{G}_{\Xi(\Theta)}(X, Y) = (U_\Theta(Y), U_\Theta(X))$.*

*Proof.* We have to show $\mathcal{G}'_{\Xi(\Theta)}(X, Y) = U_\Theta(Y)$. Recall that $\Xi(\Theta) = (A, R, C^{in})$, where $C_a^{in} = \{\emptyset\}$ for each $a \in A$. Thus for any argument $a \in A$, we find that $par(a) = Attackers_\Theta(a)$. Now

$$\begin{aligned} a \in \mathcal{G}'_{\Xi(\Theta)}(X, Y) & \text{ iff } B \in C_a^{in}, B \subseteq X, (par(a) \setminus B) \cap Y = \emptyset \\ & \text{ iff } B = \emptyset, B \subseteq X, (par(a) \setminus B) \cap Y = \emptyset \\ & \text{ iff } par(a) \cap Y = \emptyset \\ & \text{ iff } Attackers_\Theta(a) \cap Y = \emptyset \\ & \text{ iff } a \in U_\Theta(Y) \end{aligned}$$

□

In the remainder, we will denote the four-valued approximation operator of an argumentation framework $\Theta$ by $\mathcal{F}_\Theta$; we formally define $\mathcal{F}'_\Theta \overset{\text{def}}{=} \mathcal{G}'_{\Xi(\Theta)}$. It follows by definition that the characteristic operator $\mathcal{F}_\Theta$ of an AF is its own stable operator:

**Lemma 4.2.** *For any argumentation framework $\Theta$, we have $\mathcal{SF}_\Theta = \mathcal{F}_\Theta$.*

*Proof.* Let $\Theta = (A, R)$ and $X, Y \subseteq A$. We have to show $\mathcal{SF}'_\Theta(X, Y) = \mathcal{F}'_\Theta(X, Y)$. Now $\mathcal{SF}'_\Theta(X, Y) = lfp(\mathcal{F}'_\Theta(\cdot, Y)) = lfp(U_\Theta(Y)) = U_\Theta(Y) = \mathcal{F}'_\Theta(X, Y)$. □

This means informally that (in a sense) there are fewer semantics for Dung frameworks than there are for ADFs, logic programming, default logic and autoepistemic logic. Translated into logic programming language, we have that in Dung-style argumentation, supported and stable models coincide, and well-founded semantics equals Kripke-Kleene semantics. Put in different terms of default and autoepistemic logics: for argumentation frameworks, Moore expansions and Reiter extensions coincide!

In principle, this collapsing picture could be due to a mistake in our definition of the characteristic operator. In the following section, it will become clear that this is not the case and the characteristic operator of an argumentation framework is well-designed: we show next how the major semantics of argumentation frameworks can be redefined in terms of fixpoints of the characteristic operator.

## 4.1 Fixpoint Semantics for Abstract Argumentation Frameworks

As a first illustration of universality of the characteristic operator of an AF, we recapitulate a result that is well-known in the argumentation community: the operator $U_\Theta$ (which is at the heart of $\mathcal{F}_\Theta$) can emulate the characteristic function $F_\Theta$ of an argumentation framework.

**Lemma 4.3** ([9, Lemma 45])**.** *For any AF $\Theta$, we have $F_\Theta = U_\Theta^2$.*

For our operator $\mathcal{F}_\Theta$, this means that for any $X, Y \subseteq A$ we have

$$\mathcal{F}_\Theta^2(X,Y) = \mathcal{F}_\Theta(U_\Theta(Y), U_\Theta(X)) = (U_\Theta^2(X), U_\Theta^2(Y)) = (F_\Theta(X), F_\Theta(Y))$$

There are several works in the literature that redefine argumentation semantics in terms of (pre-/post-)fixpoints of the two operators $F_\Theta$ and $U_\Theta$ [1, 17]. Since the two operators are closely related and the characteristic approximating operator $\mathcal{F}_\Theta$ can express them both, we can reconstruct argumentation semantics based entirely on this single operator.

We begin with the simplest semantics: recall that for $\Theta = (A,R)$ a set $E$ of arguments is a stable extension iff $E = U_\Theta(E)$, so the following is immediate.

**Proposition 4.4.** *Let $\Theta = (A,R)$ be an argumentation framework and $E \subseteq A$. Then $E$ is a stable extension of $\Theta$ iff $\mathcal{F}_\Theta(E,E) = (E,E)$.*

It is almost as easy to characterise the class of complete extensions:

**Proposition 4.5.** *Let $\Theta = (A,R)$ be an argumentation framework and $E \subseteq A$. Then $E$ is a complete extension of $\Theta$ iff for some $E' \subseteq A$ the pair $(E,E')$ is a consistent fixpoint of $\mathcal{F}_\Theta$.*

*Proof.*

$$\begin{aligned}
&\text{There is an } E' \subseteq A \text{ with } E \subseteq E' \text{ and } \mathcal{F}_\Theta(E,E') = (E,E')\\
\text{iff } & E \subseteq E' \text{ and } E' = U_\Theta(E) \text{ and } E = U_\Theta(E')\\
\text{iff } & E \subseteq U_\Theta(E) \text{ and } E = U_\Theta^2(E)\\
\text{iff } & E \text{ is conflict-free and } E = F_\Theta(E)\\
\text{iff } & E \text{ is a complete extension.}
\end{aligned}$$
□

As an easy corollary, we get the grounded semantics as the $\leq_i$-least fixpoint of the characteristic operator. This fixpoint exists since $\mathcal{F}_\Theta$ is $\leq_i$-monotone.

**Corollary 4.6.** *Let $\Theta = (A,R)$ be an argumentation framework and $E \subseteq A$. Then $E$ is the grounded extension of $\Theta$ iff for some $E' \subseteq A$ the pair $(E,E')$ is the $\leq_i$-least fixpoint of $\mathcal{F}_\Theta$.*

In the sequel, we use the term "complete extension" for the set $E$ and the pair $(E, E')$ interchangeably. It follows by definition that preferred extensions are exactly those consistent fixpoints where $E$ is $\subseteq$-maximal – the M-supported models of $\mathcal{F}_\Theta$.

**Proposition 4.7.** *Let $\Theta = (A, R)$ be an argumentation framework and $E \subseteq A$. Then $E$ is a preferred extension of $\Theta$ iff for some $E' \subseteq A$ the pair $(E, E')$ is a consistent fixpoint of $\mathcal{F}_\Theta$ where $E$ is $\subseteq$-maximal.*

Alternatively, we can say that for a consistent pair $(E, E')$ the lower bound $E$ is a preferred extension if and only if the pair is M-supported/M-stable for $\mathcal{F}_\Theta$. This immediately yields a "preferred" semantics for default logic, which is an improvement upon a result by Dung [9, Theorem 43], who defined preferred semantics for default logic only through a translation to infinite AFs.

Semi-stable extensions are those complete ones where the set of arguments in the upper but not in the lower bound (the undecided arguments) is minimal – L-supported/L-stable models.

**Proposition 4.8.** *Let $\Theta = (A, R)$ be an argumentation framework and $E \subseteq A$. Then $E$ is a semi-stable extension of $\Theta$ iff for some $E' \subseteq A$ the pair $(E, E')$ is a consistent fixpoint of $\mathcal{F}_\Theta$ where $E' \setminus E$ is $\subseteq$-minimal.*

*Proof.* $E \cup \mathit{Attacked}_\Theta(E)$ is $\subseteq$-maximal iff $E \cup \overline{E'}$ is $\subseteq$-maximal iff $\overline{E \cup \overline{E'}}$ is $\subseteq$-minimal iff $\overline{E} \cap E'$ is $\subseteq$-minimal iff $E' \setminus E$ is $\subseteq$-minimal. □

Finally, we show that the ADF versions of "admissible" (Definition 3.3) and "conflict-free" [3, Definition 2] are proper generalisations of the respective AF notions. This is easily shown using their respective associated approximating operators.

**Proposition 4.9.** *Let $\Theta = (A, R)$ be an argumentation framework and $X \subseteq A$. Then $X$ is an admissible set for $\Theta$ iff $(X, U_\Theta(X))$ is an admissible pair for $\mathcal{F}_\Theta$.*

*Proof.* Abbreviate $Y \stackrel{\text{def}}{=} U_\Theta(X)$. We have the following equivalences:

$$\begin{aligned}
&X \text{ is an admissible set for } \Theta\\
\text{iff } &X \text{ is conflict-free and } X \subseteq F_\Theta(X)\\
\text{iff } &X \subseteq U_\Theta(X) \text{ and } X \subseteq U_\Theta^2(X)\\
\text{iff } &X \subseteq Y \text{ and } X \subseteq U_\Theta(Y) \text{ and } Y \subseteq U_\Theta(X) \text{ and } U_\Theta(X) \subseteq Y\\
\text{iff } &X \subseteq Y, (X, Y) \leq_t (U_\Theta(Y), U_\Theta(X)) \text{ and } (X, Y) \leq_i (U_\Theta(Y), U_\Theta(X))\\
\text{iff } &(X, Y) \text{ is consistent, } (X, Y) \leq_t \mathcal{F}_\Theta(X, Y) \text{ and } (X, Y) \leq_i \mathcal{F}_\Theta(X, Y)\\
\text{iff } &(X, Y) \text{ is an admissible pair for } \mathcal{F}_\Theta
\end{aligned}$$

□

It works equally easily in the case of conflict-freeness:

**Proposition 4.10.** *Let $\Theta = (A, R)$ be an argumentation framework and $X \subseteq A$. Then $X$ is conflict-free in $\Theta$ iff for some $Y$ with $X \subseteq Y \subseteq U_\Theta(X)$ we have $(X, Y) \leq_t \mathcal{F}_\Theta(X, Y)$.*

*Proof.*

$$\begin{aligned}
X \text{ is conflict-free in } \Theta \text{ iff } & X \subseteq U_\Theta(X) \\
\text{iff } & X \subseteq Y \subseteq U_\Theta(X) \text{ for some } Y \\
\text{iff } & X \subseteq U_\Theta(Y) \subseteq U_\Theta(X) \text{ and } Y \subseteq U_\Theta(X) \\
\text{iff } & X \subseteq \mathcal{F}'_\Theta(X,Y) \subseteq \mathcal{F}''_\Theta(X,Y) \text{ and } Y \subseteq \mathcal{F}''_\Theta(X,Y) \\
\text{iff } & X \subseteq \mathcal{F}'_\Theta(X,Y) \text{ and } Y \subseteq \mathcal{F}''_\Theta(X,Y) \\
\text{iff } & (X,Y) \leq_t \mathcal{F}_\Theta(X,Y)
\end{aligned}$$

□

We have seen how all of the semantical notions for AFs considered in this paper can be recast in terms of the approximating operator of an AF, as fixpoints, or post- and prefixpoints with respect to the information ordering $\leq_i$ and/or the truth ordering $\leq_t$. This tells us that operators associated with an argumentation framework are useful tools to study the semantics of the AF. This technique of associating operators with a knowledge base and then studying the operators to study the knowledge base is successfully and widely used in logic programming. In the next section, we will see that this enables us to elegantly build a bridge from abstract argumentation to logic programming via the approximation operators associated with the respective objects of study.

## 4.2 From Argumentation Frameworks to Logic Programs

There are different translations from AFs into LPs in the literature: the one we call the standard translation, and the one devised by Dung [9] to demonstrate how logic programs could be used to implement abstract argumentation. We consider each of the translations in turn and lastly show that they produce equivalent logic programs.

### 4.2.1 Standard Translation

The translation we refine below was introduced as "well-known" in Gabbay and d'Avila Garcez [13, Example 1.2]. They do not provide a definition or motivation for that translation, but our subsequent results will show that the intuition behind it is sound and the name "standard translation" is justified. The standard logic program resulting from an AF uses the set of arguments as its underlying signature. A rule is created for each argument $a$, and the rule basically says "$a$ is accepted if none of its attackers is accepted."

Since AFs are in particular ADFs, the standard logic program of an AF $\Theta$ is given by $\Pi(\Xi(\Theta))$, that is, translating the AF $\Theta$ into an ADF $\Xi(\Theta)$ and that further into the standard LP of the ADF. For AFs $\Theta = (A, R)$, the definition of its standard logic program can be simplified to the following:

$$\Pi(\Theta) \stackrel{\text{def}}{=} \{a \leftarrow \mathit{not}\, \mathit{Attackers}_\Theta(a) \mid a \in A\}$$

Note that the positive body is empty in general since there is no notion of support in classical Dung-style AFs. Also, the negative bodies of the rules are finite if and only if the framework is finitary.

It should be noted that the standard translation from AFs to LPs is not modular, since the LP rule for an atom $a$ depends on all attackers of $a$. This might seem paradoxical at first, since the standard translation from ADFs to LPs *is* modular with respect to statements. But recall that the union of two ADFs is defined whenever the two have disjoint statements, so for AFs with disjoint sets of arguments the standard translation is again modular with respect to arguments.

It is immediate from Lemma 3.13 that the associated operators of AFs $\Theta$ and their translated logic program $\Pi(\Theta)$ are the same.

**Corollary 4.11.** *For any argumentation framework $\Theta$, we have $\mathcal{F}_\Theta = \mathcal{T}_{\Pi(\Theta)}$.*

Now we know from Lemma 4.2 that the approximation operator of any AF $\Theta$ is its own stable operator – in symbols $\mathcal{F}_\Theta = \mathcal{SF}_\Theta$. Combining these two results about $\mathcal{F}_\Theta$ leads to the following lemma, which nicely pictures the special role of argumentation frameworks in the realm of nonmonotonic reasoning formalisms.

**Lemma 4.12.** *For any AF $\Theta$, we have $\mathcal{T}_{\Pi(\Theta)} = \mathcal{F}_\Theta = \mathcal{SF}_\Theta = \mathcal{ST}_{\Pi(\Theta)}$.*

Since the consequence operator of a logic program yields its Kripke-Kleene and well-founded models as well as its two-valued and three-valued supported and stable models, this lemma immediately gives rise to several important coincidence results, accumulated in the first main result of this section below. Its first and last items are obvious. The second item contains the conclusion of Wu et al. [24, Theorem 17] (they did not look at supported semantics), while the third and fourth items imply new results that solve open problems posed there.

**Theorem 4.13.** *Let $\Theta$ be an AF. The following are identical:*

1. *the grounded extension of $\Theta$, the Kripke-Kleene model of $\Pi(\Theta)$ and the well-founded model of $\Pi(\Theta)$;*
2. *complete extensions of $\Theta$, three-valued supported models of $\Pi(\Theta)$ and three-valued stable models of $\Pi(\Theta)$;*
3. *preferred extensions of $\Theta$, M-supported models of $\Pi(\Theta)$ and M-stable models of $\Pi(\Theta)$;*
4. *semi-stable extensions of $\Theta$, L-supported models of $\Pi(\Theta)$ and L-stable models of $\Pi(\Theta)$;*
5. *stable extensions of $\Theta$, two-valued supported models of $\Pi(\Theta)$ and two-valued stable models of $\Pi(\Theta)$.*

*Proof.* The first item is obvious, since they are the least fixpoint of the same operator; the rest follows from Lemma 4.12 and Propositions 4.5, 4.7, 4.8 and 4.4. □

As witnessed by Lemma 3.13, for the standard translation the correspondence between AFs and LPs is immediate. We will next consider a different translation where this correspondence is less obvious, albeit still present. Most importantly, that translation will be modular for all argumentation frameworks.

### 4.2.2 Dung's Translation

Dung duplicates the arguments, thereby explicitly keeping track of their being *in* or *out*: for $a \in A$, a new propositional variable $\text{-}a$ expresses defeat of $a$ by some counterargument. Note that this translation is modular with respect to both arguments and attacks, and furthermore rule bodies are always finite.[5]

**Definition 4.1.** Let $\Theta = (A, R)$ be an argumentation framework. Define $\text{-}A \overset{\text{def}}{=} \{\text{-}a \mid a \in A\}$, $A^\pm \overset{\text{def}}{=} A \cup \text{-}A$ and a logic program over $A^\pm$ as follows.

$$\Pi_D(\Theta) \overset{\text{def}}{=} \{a \leftarrow \mathit{not}\,\text{-}a \mid a \in A\} \cup \{\text{-}a \leftarrow b \mid (b, a) \in R\}$$

[5] Dung's original translation is slightly different; he uses a first-order signature and logic program atoms with variables [9]. Definition 4.1 above is merely a syntactical variant that already incorporates ground instantiation.

Intuitively, an argument $a$ is accepted (signified by atom $a$) unless it is defeated (signified by atom $\text{-}a$). An argument is defeated if it is attacked by an accepted argument.

We show next that the four-valued one-step consequence operator for the logic program resulting from Dung's translation essentially does the same as the characteristic operator of the original argumentation framework. It only needs twice as many steps due to the syntactical duplication of arguments.

To show this result, we need the technical notion of coherence: in words, a pair is coherent if it respects the intuition of $\text{-}a$ for $a \in A$, in the sense that $a$ is true iff $\text{-}a$ is false and vice versa. A pair $(S, P)$ of sets of arguments can be extended to matching pairs $(S^*, P^*)$ of logic program atoms over $A^{\pm}$ in a straightforward way.

**Definition 4.2.** Let $A$ be a set of arguments and $S^*, P^* \subseteq A^{\pm}$. The pair $(S^*, P^*)$ is *coherent* iff for all $a \in A$, we find $a \in S^*$ iff $\text{-}a \notin P^*$ and $a \in P^*$ iff $\text{-}a \notin S^*$. For $S, P \subseteq A$, define $co(S, P) \overset{\text{def}}{=} (S \cup \text{-}\overline{P}, P \cup \text{-}\overline{S})$.[6]

Observe that $\text{-}\overline{X} = \{\text{-}a \mid a \notin X\}$, so it is clear that the pair $co(S, P)$ is coherent. What the function does, intuitively, is simple: if $a$ is not in the upper bound $P$, that is, cannot become true any more, then it can be considered false, which is expressed by adding $\text{-}a$ to the lower bound; likewise, if $a$ is not in the lower bound $S$, that is, is not yet considered true, then its falsity must be considered an option, which leads to $\text{-}a$ being added to the upper bound. These manipulations are entirely syntactic and do not mention attacks.

We are now ready to show that for an AF $\Theta = (A, R)$, its standard translation $\Pi(\Theta)$ and Dung translation $\Pi_D(\Theta)$ have the same four-valued supported models with respect to the original signature $A$. Technically, we show that the fixpoints of their four-valued one-step consequence operators coincide.

**Theorem 4.14.** *Let $\Theta = (A, R)$ be an argumentation framework with standard translation $\Pi$ and Dung-translation $\Pi_D$ and let $S, P \subseteq A$.*

$$\mathcal{T}_{\Pi}(S, P) = (S, P) \quad \textit{iff} \quad \mathcal{T}_{\Pi_D}(co(S, P)) = co(S, P)$$

*Proof.* We first observe that for any $X, Y \subseteq A$ and $a \in A$, by definition of $\Pi_D$ we have $a \in \mathcal{T}'_{\Pi_D}(X, Y)$ iff $\text{-}a \notin Y$ and $\text{-}a \in \mathcal{T}'_{\Pi_D}(X, Y)$ iff $a \notin U_{\Theta}(X)$, whence $\mathcal{T}'_{\Pi_D}(X, Y) = \{a \mid \text{-}a \notin Y\} \cup \text{-}\overline{U_{\Theta}(X)}$ and $\mathcal{T}'_{\Pi_D}(X, \text{-}\overline{Y}) = Y \cup \text{-}\overline{U_{\Theta}(X)}$. Now

$$\begin{aligned}
&\mathcal{T}_{\Pi}(S, P) = (S, P) \\
\text{iff } &\mathcal{F}_{\Theta}(S, P) = (S, P) \\
\text{iff } &(U_{\Theta}(P), U_{\Theta}(S)) = (S, P) \\
\text{iff } &S = U_{\Theta}(P) \text{ and } P = U_{\Theta}(S) \\
\text{iff } &S = U_{\Theta}(P) \text{ and } P = U_{\Theta}(S) \text{ and } \text{-}\overline{U_{\Theta}(P)} = \text{-}\overline{S} \text{ and } \text{-}\overline{U_{\Theta}(S)} = \text{-}\overline{P} \\
\text{iff } &S \cup \text{-}\overline{U_{\Theta}(S)} = S \cup \text{-}\overline{P} \text{ and } P \cup \text{-}\overline{U_{\Theta}(P)} = P \cup \text{-}\overline{S} \\
\text{iff } &(S \cup \text{-}\overline{U_{\Theta}(S)}, P \cup \text{-}\overline{U_{\Theta}(P)}) = (S \cup \text{-}\overline{P}, P \cup \text{-}\overline{S}) \\
\text{iff } &(\mathcal{T}'_{\Pi_D}(S, \text{-}\overline{S}), \mathcal{T}'_{\Pi_D}(P, \text{-}\overline{P})) = (S \cup \text{-}\overline{P}, P \cup \text{-}\overline{S}) \\
\text{iff } &(\mathcal{T}'_{\Pi_D}(S \cup \text{-}\overline{P}, P \cup \text{-}\overline{S}), \mathcal{T}'_{\Pi_D}(P \cup \text{-}\overline{S}, S \cup \text{-}\overline{P})) = (S \cup \text{-}\overline{P}, P \cup \text{-}\overline{S}) \\
\text{iff } &\mathcal{T}_{\Pi_D}(S \cup \text{-}\overline{P}, P \cup \text{-}\overline{S}) = (S \cup \text{-}\overline{P}, P \cup \text{-}\overline{S}) \\
\text{iff } &\mathcal{T}_{\Pi_D}(co(S, P)) = co(S, P)
\end{aligned}$$

□

[6] The notation is entirely unambiguous since for any $S \subseteq A$ we have $\text{-}\overline{S} = \overline{\text{-}S}$.

Furthermore, coherent pairs are also the *only* fixpoints of $\mathcal{T}_{\Pi_D}$.

**Proposition 4.15.** *Let $\Theta = (A, R)$ be an AF, $\Pi_D$ be its Dung translation over $A^{\pm}$ and let $S^*, P^* \subseteq A^{\pm}$. If $\mathcal{T}_{\Pi_D}(S^*, P^*) = (S^*, P^*)$, then $(S^*, P^*)$ is coherent.*

*Proof.* Let $\mathcal{T}_{\Pi_D}(S^*, P^*) = (S^*, P^*)$. Now $S^* = \{a \mid \text{-}a \notin P^*\} \cup \text{-}\overline{U_\Theta(S^*)}$ and $P^* = \{a \mid \text{-}a \notin S^*\} \cup \text{-}\overline{U_\Theta(P^*)}$. For $a \in A$, it immediately follows that $a \in S^*$ iff $\text{-}a \notin P^*$ and $a \in P^*$ iff $\text{-}a \notin S^*$, thus $(S^*, P^*)$ is coherent. □

Hence for any semantics derived from the operator $\mathcal{T}_{\Pi_D}$ which is only "interested" in atoms from $A$, the choice between standard translation and Dung translation is semantically inessential. We remark that Dung's translation has the advantage of producing a logic program where each rule has a finite body.

Theorem 4.14 and Proposition 4.15 immediately yield the same nice correspondence picture from the standard translation (Theorem 4.13) for Dung's translation. The first and last items below are again obvious for our setting, parts of them were also proved by Dung [9, Theorem 62]. Correspondence results 2, 3 and 4 are completely new.

**Theorem 4.16.** *Let $\Theta = (A, R)$ be an argumentation framework. The following are in one-to-one correspondence:*

1. *the grounded extension of $\Theta$, the Kripke-Kleene model of $\Pi_D(\Theta)$ and the well-founded model of $\Pi_D(\Theta)$;*
2. *complete extensions of $\Theta$, three-valued supported models of $\Pi_D(\Theta)$ and three-valued stable models of $\Pi_D(\Theta)$;*
3. *preferred extensions of $\Theta$, M-supported models of $\Pi_D(\Theta)$ and M-stable models of $\Pi_D(\Theta)$;*
4. *semi-stable extensions of $\Theta$, L-supported models of $\Pi_D(\Theta)$ and L-stable models of $\Pi_D(\Theta)$;*
5. *stable extensions of $\Theta$, two-valued supported models of $\Pi_D(\Theta)$ and two-valued stable models of $\Pi_D(\Theta)$.*

*Proof.* Follows from Theorem 4.14, Proposition 4.15 and Propositions 4.5, 4.7, 4.8 and 4.4. □

This theorem conclusively shows that Dung's modular translation from AFs to LPs is faithful with respect to all operator-based semantics. We infer that propositional normal logic programs are at least as expressive as abstract argumentation frameworks.

## 4.3 From Logic Programs to Argumentation Frameworks

For ADFs, we have seen how the standard translation into logic programs could straightforwardly be reversed into a translation from normal logic programs to ADFs that was sound with respect to both two-valued supported and stable model semantics. In the case of AFs, however, things are different. The standard logic programs obtained from Dung argumentation frameworks are of a very specific form: there is one rule for each atom, and there are no positive atoms in rule bodies. For logic programs from this class, it is of course clear how to transform them into argumentation frameworks. For programs outside of this class, however, the standard translation does not give us any clues on how to proceed.

Although it is certainly possible to devise semantics-dependent translations from logic programs into argumentation frameworks (as a start, consider translating a logic program into an ADF to which in turn the translation from Brewka et al. [4] is applied), we consider it unlikely that any such translation is polynomial, faithful *and* modular. In particular, it is highly unlikely that a polynomial and modular translation is faithful with respect to both supported *and* stable semantics, as these two semantics are not equal in general but coincide for abstract argumentation frameworks. A more extensive investigation of this subject would certainly be interesting, but also involve general considerations about argumentation semantics and fall beyond the scope of this paper.

# 5 General Semantics for Approximating Operators

We have seen how the characteristic operator of an ADF can be used to redefine the existing ADF semantics. In addition, this introduced the admissible, preferred and stable semantics for all ADFs – they were previously only defined for bipolar ADFs. We have also seen that an ADF $\Xi$ and its standard logic program $\Pi(\Xi)$ correspond on all semantics which are defined for both ADFs and LPs. Finally, we have seen how the characteristic operator of Dung-style argumentation frameworks (given by AF-based ADFs) allows to redefine AF semantics for operators. This allows us to easily transfer definitions of semantics from abstract argumentation to abstract dialectical frameworks, logic programming and beyond – to the general case of approximating operators.

## 5.1 Conflict-free

In classical abstract argumentation, a set of arguments is conflict-free if there are no attacks amongst its members. For abstract dialectical frameworks, a set of statements is conflict-free if each statement – informally speaking – has a reason to be in the set. This reason for one entails absence of any attackers as well as presence of supporters in case the statement's acceptance conditions so requires. In operator terms, we have seen that conflict-freeness can be cast as being a postfixpoint of an operator with respect to the truth value ordering $\leq_t$. This is the four-valued generalisation of the hitherto two-valued notion of conflict-freeness.

**Definition 5.1.** Let $(L, \sqsubseteq)$ be a complete lattice and $\mathcal{O}$ an approximating operator on the bilattice $(L^2, \leq_i)$. A consistent pair $(x, y) \in L^2$ is *conflict-free for* $\mathcal{O}$ iff $(x, y) \leq_t \mathcal{O}(x, y)$.

Note that conflict-freeness requires consistency. Intuitively, an approximation $(x, y)$ is conflict-free if it is consistent (i.e. approximates at least one element) and applying the approximation operator will lead to a pair $(\hat{x}, \hat{y})$ where the new bounds are comparable to the old ones and never decrease, i.e. $x \sqsubseteq \hat{x}$ and $y \sqsubseteq \hat{y}$. Since $\mathcal{O}$ is approximating, the pair $(\hat{x}, \hat{y})$ will again be consistent.

## 5.2 Admissible

In Dung argumentation frameworks, a set of arguments is admissible if it is conflict-free and defends itself against all attacks. For abstract dialectical frameworks, we have seen in Definition 3.3 and Proposition 4.9 that a suitable ADF generalisation of AF admissibility is given by the above notion of conflict-freeness and a property that DMT call "reliability with respect to an operator" [8]. In operator-based language, an admissible pair could also be called a consistent, conflict-free pair that is a postfixpoint with respect to the information ordering $\leq_i$. For the sake of completeness we have included the following formal definition.

**Definition 5.2.** Let $(L, \sqsubseteq)$ be a complete lattice and $\mathcal{O}$ an approximating operator on the bilattice $(L^2, \leq_i)$. A consistent pair $(x, y) \in L^2$ is *admissible for* $\mathcal{O}$ iff $(x, y) \leq_t \mathcal{O}(x, y)$ and $(x, y) \leq_i \mathcal{O}(x, y)$.

Alternatively, we could say that a consistent pair $(x, y)$ is admissible iff $x \sqsubseteq \mathcal{O}'(x, y)$ and $\mathcal{O}''(x, y) = y$. The definition we chose makes it obvious that an admissible pair is always conflict-free. Additionally, any pair that is admissible for $\mathcal{O}$ is also $\mathcal{O}$-reliable [8]. DMT point out that $\mathcal{O}$-reliable pairs – consistent pairs whose $\mathcal{O}$-revisions are at least as accurate – are especially useful for studying fixpoints of $O$, the original operator that $\mathcal{O}$ approximates. In particular, the $\leq_i$-least element $(\bot, \top)$ is $\mathcal{O}$-reliable; iterating $\mathcal{O}$ on it leads to the Kripke-Kleene semantics, which provides a more precise approximation of all fixpoints of the approximated operator $O$.

## 5.3 Semi-stable

Theorem 4.13 and Proposition 4.8 immediately yield a definition of L-stable/semi-stable semantics for default and autoepistemic logics. Complete semantics for the two are given by consistent fixpoints (those $(x, y)$ with $x \sqsubseteq y$) of an approximating operator. To generalise semi-stable to operators we simply have to generalise the minimality criterion of L-stable models for logic programming. Since this involves algebraic operations on lattice elements, we have to make some more restricting assumptions on the underlying lattice.

In the sequel, for a complete lattice $(L, \sqsubseteq)$ with join $\sqcup$ and meet $\sqcap$, we assume the existence of a function $\cdot^{-1} : L \to L$ such that for any $x, y \in L$,

- $\left(x^{-1}\right)^{-1} = x$ ($\cdot^{-1}$ is involutive)
- $(x \sqcup y)^{-1} = x^{-1} \sqcap y^{-1}$ and
- $(x \sqcap y)^{-1} = x^{-1} \sqcup y^{-1}$ (de Morgan's laws)

In the special cases we have seen so far, the role of this "negation" is played by set complement with respect to the underlying vocabulary.

**Definition 5.3.** Let $(L, \sqsubseteq)$ be a complete lattice and $\mathcal{O}$ an approximating operator on its corresponding bilattice $(L^2, \leq_i)$. A consistent pair $(x, y)$ is *L-supported* iff it is a fixpoint of $\mathcal{O}$ and $y \sqcap x^{-1}$ is $\sqsubseteq$-minimal. A consistent pair $(x, y)$ is *L-stable* iff it is a fixpoint of $\mathcal{SO}$ and $y \sqcap x^{-1}$ is $\sqsubseteq$-minimal.

For the special case of argumentation, these general definitions of L-supported and L-stable reduce to a consistent fixpoint $(S, P)$ of $\mathcal{F}_\Theta = \mathcal{SF}_\Theta$ such that $P \cap \overline{S} = P \setminus S$ (the set of undecided arguments) is $\subseteq$-minimal – a semi-stable extension.

## 5.4 Stage

We now turn to a semantics that is not based on admissibility, but only on conflict-freeness: stage extensions. Recall that a set $S \subseteq A$ is a stage extension of $\Theta = (A, R)$ iff it is conflict-free and has maximal range, that is, the set $S \cup \mathit{Attacked}_\Theta(S)$ is $\subseteq$-maximal [23]. Alternatively, stage semantics can be seen as a less restrictive version of semi-stable semantics where "admissible" is replaced by "conflict-free." This characterisation, in effect, will lead to our operator generalisation of stage semantics.

**Definition 5.4.** Let $\Xi = (S, L, C)$ be an abstract dialectical framework and $X \subseteq Y \subseteq S$. The consistent pair $(X, Y)$ is a *stage pair of* $\Xi$ iff $(X, Y)$ is conflict-free and $Y \setminus X$ is $\subseteq$-minimal.

We next prove that this definition of stage extensions is indeed a generalisation of the notion for AFs.

**Proposition 5.1.** *Let $\Theta = (A, R)$ be an AF and $\Xi = \Xi(\Theta)$ be its associated ADF. A set $X \subseteq A$ is a stage extension of $\Theta$ iff the pair $(X, U_\Theta(X))$ is a stage pair of $\Xi$.*

*Proof.* Abbreviate $Y \overset{\text{def}}{=} U_\Theta(X)$. We have the following equivalences:

$$\begin{aligned}
&X \text{ is a stage extension for } \Theta\\
\text{iff } &X \text{ is conflict-free and } X \cup \mathit{Attacked}_\Theta(X) \text{ is } \subseteq\text{-maximal}\\
\text{iff } &X \text{ is conflict-free and } X \cup \overline{U_\Theta(X)} \text{ is } \subseteq\text{-maximal}\\
\text{iff } &X \subseteq U_\Theta(X) \text{ and } \overline{X} \cap U_\Theta(X) \text{ is } \subseteq\text{-minimal}\\
\text{iff } &X \subseteq Y \text{ and } Y \setminus X \text{ is } \subseteq\text{-minimal}\\
\text{iff } &X \subseteq U_\Theta(Y) \subseteq U_\Theta(X) = Y \subseteq U_\Theta(X) \text{ and } Y \setminus X \text{ is } \subseteq\text{-minimal}\\
\text{iff } &X \subseteq \mathcal{F}'_\Theta(X, Y) \text{ and } Y \subseteq \mathcal{F}''_\Theta(X, Y) \text{ and } Y \setminus X \text{ is } \subseteq\text{-minimal}\\
\text{iff } &(X, Y) \leq_t \mathcal{F}_\Theta(X, Y) \text{ and } Y \setminus X \text{ is } \subseteq\text{-minimal}\\
\text{iff } &(X, Y) \leq_t \mathcal{G}_\Xi(X, Y) \text{ and } Y \setminus X \text{ is } \subseteq\text{-minimal}\\
\text{iff } &(X, Y) \text{ is a stage pair of } \Xi
\end{aligned}$$

□

As usual, Definition 5.4 straightforwardly yields stage models for logic programming and stage extensions/expansions for default and autoepistemic logics, defined through stage pairs of an approximating operator $\mathcal{O}$ in a bilattice.

**Definition 5.5.** Let $(L, \sqsubseteq)$ be a complete lattice and $\mathcal{O}$ an approximating operator on its corresponding bilattice $(L^2, \leq_i)$. A consistent pair $(x, y) \in L^2$ is a *stage pair for $\mathcal{O}$* iff $(x, y) \leq_t \mathcal{O}(x, y)$ and $y \sqcap x^{-1}$ is $\sqsubseteq$-minimal.

# 6 Conclusion

We embedded abstract dialectical frameworks into Denecker et al.'s lattice-theoretical formalism for the abstract study of nonmonotonic logical languages. This provides useful insights into the relationship of abstract argumentation frameworks and abstract dialectical frameworks with other nonmonotonic knowledge representation formalisms.

In this last section, we will provide a concise overview over the results of our investigation. First, for reference and as a completion of the table in Definition 2.2, we review the definitions of operator-based semantics in Table 1.

Figure 1 then depicts the relationship between the different semantical notions explored in this paper. If a semantics $\sigma$ is seen as a function assigning to a knowledge base $\kappa$ over vocabulary $A$ a set of pairs $(X, Y)$ with $X, Y \subseteq A$, then a partial order on semantics is given by $\sigma_1 \leq \sigma_2$ iff $\sigma_1(\kappa) \subseteq \sigma_2(\kappa)$ for all $\kappa$. In the figure, an arrow from $\sigma_1$ to $\sigma_2$ expresses $\sigma_1 \leq \sigma_2$ – in words, all $\sigma_1$-models are also $\sigma_2$-models.

Next, Table 2 shows the correspondences between different argumentation semantics and operator-based semantics. The operator-based semantics lead to new semantics for default logic and autoepistemic logics via their respective consequence operators [7]. A discussion of these semantics is however out of the scope of this paper.

Finally, Figure 2 on page 34 shows the location of abstract dialectical frameworks with respect to different approaches in the area of nonmonotonic reasoning. We use a very strong notion of one formalism being at least as expressive as another: the existence of a polynomial and modular translation that is faithful with respect to all operator-based semantics. Such results existed previously for the translation from logic programs into default theories of Marek and

| | |
|---|---|
| conflict-free pair $(x, y)$ | $(x, y) \leq_t \mathcal{O}(x, y)$ |
| stage pair $(x, y)$ | $(x, y) \leq_t \mathcal{O}(x, y)$ and $y \sqcap x^{-1}$ is $\sqsubseteq$-minimal |
| admissible pair $(x, y)$ | $(x, y) \leq_t \mathcal{O}(x, y)$ and $(x, y) \leq_i \mathcal{O}(x, y)$ |
| Kripke-Kleene semantics | $\mathit{lfp}(\mathcal{O})$ |
| three-valued supported model $(x, y)$ | $\mathcal{O}(x, y) = (x, y)$ |
| M-supported model $(x, y)$ | $\mathcal{O}(x, y) = (x, y)$ and $(x, y)$ is $\leq_i$-maximal |
| L-supported model $(x, y)$ | $\mathcal{O}(x, y) = (x, y)$ and $y \sqcap x^{-1}$ is $\sqsubseteq$-minimal |
| two-valued supported model $(x, x)$ | $\mathcal{O}(x, x) = (x, x)$ |
| well-founded semantics | $\mathit{lfp}(\mathcal{SO})$ |
| three-valued stable model $(x, y)$ | $\mathcal{SO}(x, y) = (x, y)$ |
| M-stable model $(x, y)$ | $\mathcal{SO}(x, y) = (x, y)$ and $(x, y)$ is $\leq_i$-maximal |
| L-stable model $(x, y)$ | $\mathcal{SO}(x, y) = (x, y)$ and $y \sqcap x^{-1}$ is $\sqsubseteq$-minimal |
| two-valued stable model $(x, x)$ | $\mathcal{SO}(x, x) = (x, x)$ |

*Table 1: Operator-based semantical notions. All of them are defined for $x, y \in L$ with $x \sqsubseteq y$ for complete lattices $(L, \sqsubseteq)$ and approximating operators $\mathcal{O}$ on their corresponding bilattice, in some cases (stage, L-supported, L-stable) with additional restrictions on join, meet and involution operations on the lattice.*

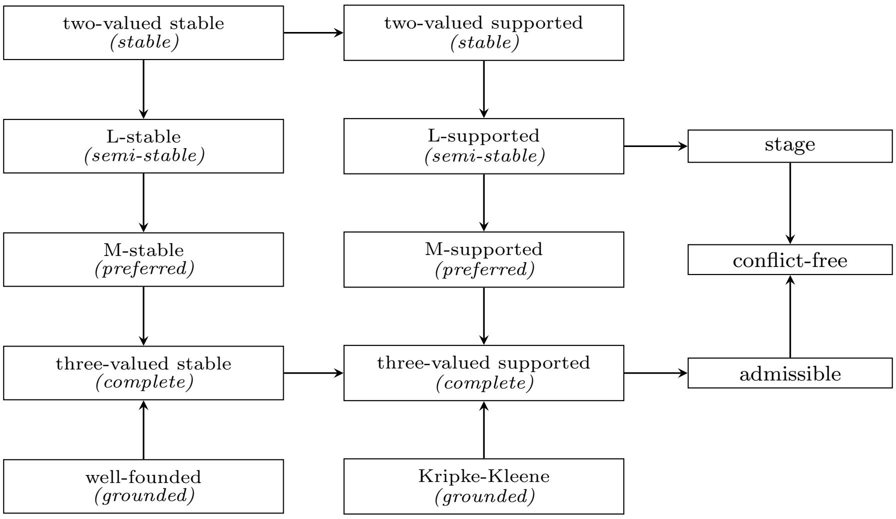

*Figure 1: Inclusion relations between operator-based semantics. Nodes depict semantical notions for elements of a bilattice, where the names in parentheses are argumentation versions of these notions. Directed edges indicate subset relationships between the sets of all bilattice elements which satisfy the respective semantical notion. For example, the arrow from admissible to conflict-free means that all admissible pairs are conflict-free.*

Truszczyński [20], and the translation from default logic into autoepistemic logic of Konolige [19] – for details see Denecker et al. [6]. In this paper, we added argumentation frameworks and abstract dialectical frameworks to the picture.

**Related work.** The several new correspondence results for AFs and logic programs we proved extended results of Wu et al. [24], who showed correspondence of complete extensions and three-valued stable models. While the results of Wu et al. [24] use the translation of Gabbay and d'Avila Garcez [13], they do not motivate the use of this – we call it standard – translation nor provide a comparison to the much older Dung translation. In this paper we showed that using

| Operator | AF | ADF |
|---|---|---|
| **conflict-free pair** | conflict-free set | conflict-free set/**pair** |
| **stage pair** | stage extension | **stage pair** |
| **admissible pair** | admissible set | **admissible pair** |
| Kripke-Kleene semantics | grounded extension | **Kripke-Kleene semantics** |
| ultimate Kripke-Kleene semantics | grounded extension | BW-well-founded model |
| three-valued supported model | complete extension | **three-valued supported model** |
| **M-supported model** | preferred extension | **M-supported model** |
| **L-supported model** | semi-stable extension | **L-supported model** |
| two-valued supported model | stable extension | (two-valued supported) model |
| well-founded semantics | grounded extension | **well-founded semantics** |
| three-valued stable model | complete extension | **three-valued stable model** |
| **M-stable model** | preferred extension | **M-stable model** |
| **L-stable model** | semi-stable extension | **L-stable model** |
| two-valued stable model | stable extension | **two-valued stable model** |

*Table 2: Overview over semantics for approximating operators, argumentation frameworks and abstract dialectical frameworks. Semantics newly defined in this paper are written in **bold font**. All extension semantics for AFs have at least two generalisations, a supported and a stable one. While most argumentation semantics already had a corresponding operator semantics, we found that conflict-free and admissible sets and stage extensions lead to new semantical notions for approximating operators. The operator-based versions of argumentation semantics then directly lead to the ADF generalisations of these semantics, most of which are newly defined in this paper. M/L-stable/supported models for operators are straightforwardly generalised notions from logic programming. Operator-based semantics then immediately lead to semantics for default logic and autoepistemic logic (not included in this presentation).*

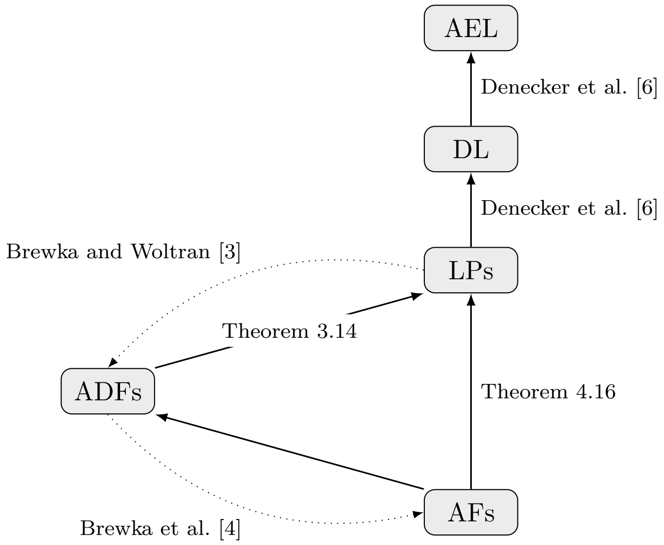

*Figure 2: Relative expressiveness of NMR formalisms. Nodes depict nonmonotonic knowledge representation formalisms; argumentation frameworks (AFs), abstract dialectical frameworks (ADFs), logic programs (LPs), default logic (DL) and autoepistemic logic (AEL), respectively. A solid directed edge expresses that there exists a polynomial, faithful and modular translation from source to target formalism, where faithful means the exact correspondence of associated approximating operators. Dotted edges denote non-modular translations which are polynomial, but only faithful with respect to two-valued (BW-)stable semantics.*

the standard translation is justified; what is more, we even proved that the standard translation and Dung's translation produce equivalent programs.

Concerning translations from AFs into LPs, related work has also been done by Egly et al. [10] – they however have a different goal: they want to efficiently implement different argumentation semantics using the stable model semantics for logic programming. Furthermore they employ meta-programming and answer set programming with variables to allow for modular translations. Besnard and Doutre [1] redefined argumentation semantics in terms of fixpoints, but they do not look at grounded or semi-stable semantics and do not use their insights to embed argumentation frameworks into the larger picture. Very recently, Grossi [17] investigated fixpoint-based definitions of argumentation semantics to study the connection between argumentation and dynamic epistemic logic. Ellmauthaler and Wallner [11] most recently provided an implementation of ADFs which is based on answer set programming.

In general, we are not aware of any works that address the relationship of abstract dialectical frameworks with other nonmonotonic knowledge representation formalisms, attempt a principled reconstruction of ADF semantics or generalise argumentation semantics to an abstract operator-based setting.

**Future work.** As we observed in Example 3.7, it is not immediately clear how to define the union of two ADFs that share statements. For specific representations of acceptance conditions, such a union should be straightforward to define; for example when using acceptance formulas, a statement's acceptance formula in the union is simply the disjunction of the acceptance formulas in the constituents. We want to devote some future work into abstracting from such specific representations and develop a general method for combining ADFs.

Corollary 3.12 has shown that Brewka and Woltran [3] defined not only the notion of an ADF model, but also the ultimate approximation of this notion. Denecker et al. [8] study several other ultimate semantics. It is an important aspect of future work to investigate these ultimate semantics in detail and to compare them with the ones investigated here.

We remarked on several occasions throughout the paper that we defined new semantics for default and autoepistemic logics (admissible, preferred, semi-stable, stage). We plan to study these semantics in greater detail, especially their strengths and weaknesses in comparison to the standard semantics of these two nonmonotonic knowledge representation formalisms.

**Acknowledgements.** The author wishes to thank Gerhard Brewka, Stefan Ellmauthaler and Johannes Peter Wallner for useful discussions and providing (counter-)examples. He is also grateful to several anonymous reviewers for providing valuable feedback on earlier versions of (parts of) this document.

# References

[1] Philippe Besnard and Sylvie Doutre. Characterization of Semantics for Argument Systems. In Didier Dubois, Christopher A. Welty, and Mary-Anne Williams, editors, *Proceedings of the Ninth International Conference on Principles of Knowledge Representation and Reasoning (KR2004)*, pages 183–193. AAAI Press, 2004.

[2] Gerhard Brewka and Thomas F. Gordon. Carneades and Abstract Dialectical Frameworks: A Reconstruction. In *Computational Models of Argument: Proceedings of COMMA 2010, Desenzano del Garda, Italy, September 8-10, 2010*, volume 216 of *Frontiers in Artificial Intelligence and Applications*, pages 3–12. IOS Press, 2010.

[3] Gerhard Brewka and Stefan Woltran. Abstract Dialectical Frameworks. In *Proceedings of KR*, pages 102–111, 2010.

[4] Gerhard Brewka, Paul E. Dunne, and Stefan Woltran. Relating the Semantics of Abstract Dialectical Frameworks and Standard AFs. In *IJCAI*, pages 780–785, 2011.

[5] Martin Caminada. On the Issue of Reinstatement in Argumentation. In *Proceedings of the Tenth European Conference on Logics in Artificial Intelligence*, volume 4160 of *Lecture Notes in Computer Science*, pages 111–123. Springer, September 2006.

[6] Marc Denecker, Victor Marek, and Miroslaw Truszczynski. Approximations, Stable Operators, Well-Founded Fixpoints and Applications in Nonmonotonic Reasoning. In *Logic-Based Artificial Intelligence*, pages 127–144. Kluwer Academic Publishers, 2000.

[7] Marc Denecker, V. Wiktor Marek, and Miroslaw Truszczynski. Uniform Semantic Treatment of Default and Autoepistemic Logics. *Artificial Intelligence*, 143(1):79–122, 2003.

[8] Marc Denecker, Victor W. Marek, and Miroslaw Truszczynski. Ultimate approximation and its application in nonmonotonic knowledge representation systems. *Information and Computation*, 192(1):84–121, 2004.

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

[10] Uwe Egly, Sarah Alice Gaggl, and Stefan Woltran. Answer-set Programming Encodings for Argumentation Frameworks. *Argument and Computation*, 1(2):147–177, 2010.

[11] Stefan Ellmauthaler and Johannes Peter Wallner. Evaluating Abstract Dialectical Frameworks with ASP. In *COMMA*, pages 505–506, 2012.

[12] Melvin Fitting. Fixpoint Semantics for Logic Programming: A Survey. *Theoretical Computer Science*, 278(1–2):25–51, 2002.

[13] Dov M. Gabbay and Artur S. d'Avila Garcez. Logical Modes of Attack in Argumentation Networks. *Studia Logica*, 93(2–3):199–230, 2009.

[14] Thomas F. Gordon and Douglas Walton. Proof Burdens and Standards. In Iyad Rahwan and Guillermo Simari, editors, *Argumentation in Artificial Intelligence*, pages 239—258. Springer, 2009.

[15] Thomas F. Gordon, Henry Prakken, and Douglas Walton. The Carneades model of argument and burden of proof. *Artificial Intelligence*, 171(10–15):875–896, 2007.

[16] Georg Gottlob. Translating Default Logic into Standard Autoepistemic Logic. *Journal of the ACM*, 42(4):711–740, 1995.

[17] Davide Grossi. Fixpoints and Iterated Updates in Abstract Argumentation. In *Proceedings of KR*, pages 65–74. AAAI Press, 2012.

[18] Tomi Janhunen. On the Intertranslatability of Non-monotonic Logics. *Annals of Mathematics and Artificial Intelligence*, 27(1–4):79–128, 1999.

[19] Kurt Konolige. On the Relation Between Default and Autoepistemic Logic. *Artificial Intelligence*, 35(3):343–382, 1988.

[20] V. Wiktor Marek and Miroslaw Truszczyński. Stable semantics for logic programs and default theories. In *North American Conference on Logic Programming*, pages 243–256. The MIT Press, 1989.

[21] Domenico Saccà and Carlo Zaniolo. Deterministic and Non-Deterministic Stable Models. *Journal of Logic and Computation*, 7(5):555–579, 1997.

[22] Alfred Tarski. A Lattice-Theoretical Fixpoint Theorem and Its Applications. *Pacific Journal of Mathematics*, 5(2):285–309, 1955.

[23] Bart Verheij. Two approaches to dialectical argumentation: admissible sets and argumentation stages. In J.-J. Ch. Meyer and L.C. van der Gaag, editors, *Proceedings of the Eighth Dutch Conference on Artificial Intelligence (NAIC'96)*, pages 357–368, 1996.

[24] Yining Wu, Martin Caminada, and Dov M. Gabbay. Complete Extensions in Argumentation Coincide with 3-Valued Stable Models in Logic Programming. *Studia Logica*, 93(2–3):383–403, 2009.