Approximating Operators and Semantics for Abstract Dialectical Frameworks

Hannes Strass*

Computer Science Institute, Leipzig University
Augustusplatz 10, 04109 Leipzig, Germany

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.

Keywords: abstract dialectical frameworks, abstract argumentation frameworks, logic programming, fixpoint semantics, approximations, nonmonotonic reasoning

1. Introduction

In recent years, abstract argumentation frameworks (AFs) [14] 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, their expressive capabilities are somewhat limited, as has been recognised many times in the literature: often it is inadequate to model argumentation scenarios having as only means of expression arguments *attacking* each other. There have been several proposals towards generalising AFs. To cite only a few examples, Prakken and Sartor [34] add *priorities* amongst arguments that are constructed from prioritised logic programming rules; Nielsen and Parsons [30] introduced attacks from *sets* of arguments; Cayrol and Lagasquie-Schiex [9] presented bipolar argumentation frameworks, in which arguments can also *support* each other; and Modgil [28] proposed attacks on *attacks* with the aim of reasoning about preferences on the object level.

*Corresponding author
Email address: strass@informatik.uni-leipzig.de (Hannes Strass)

As a general way to overcome the restrictions of Dung's AFs while staying on the abstract level, Brewka and Woltran [3] introduced abstract dialectical frameworks (ADFs). Just like AFs, these ADFs treat arguments (called *statements* there) as abstract, atomic entities whose contents are not further analysed. But instead of expressing for an argument only its attackers, ADFs associate with each statement an *acceptance condition* that determines the acceptance status of a statement given the acceptance status of its parent statements. These parents are the statements which have a say on whether the statement in question can or must (not) be accepted. In this way, AFs are recovered in the language of ADFs by specifying for each statement the acceptance condition "accept if and only if none of the attackers is accepted."

The abstract nature of Dung's AFs makes them well-suited as a target language for translations from more expressive formalisms. To be more precise, it is common to use expressive languages to model more concrete (argumentation) scenarios, and to provide these original expressive languages with semantics by translating them into Dung AFs [8, 44, 33, 40]. However, Caminada and Amgoud [8] observed that it is not always immediately clear how such translations into AFs should be defined, even for a fairly simple source formalism. A major problem that they encountered were unintended conclusions that indirectly led to inconsistency. In the same paper, Caminada and Amgoud also proposed solutions to these problems, where during translation additional precautions have to be taken to avoid undesired anomalies. Let us explain in more detail what this means in general for abstractions among knowledge representation (KR) languages.

First of all, by an abstraction we mean a translation between languages that may disregard some information. Instantiating an abstract language is then the process of translating a more concrete, more expressive language into the abstract, less expressive language. This entails that there is no dichotomy "knowledge representation language vs. abstraction formalism" – any KR language abstracts to a greater or lesser extent, and can thus be used for abstraction purposes. Whether any specific language is to be used for direct, concrete representation or for abstraction of another language depends entirely on the application domain at hand.

Naturally, we are interested in those abstractions that preserve the meaning of translated language elements in some sense. As an example, consider the language $\{yes, no\}$. It is very simple and can abstract from any decision problem whatsoever. Furthermore it is trivial to devise an intuitively correct semantics for it. But to faithfully instantiate this language to a particular decision problem – say, the satisfiability problem of propositional logic –, the problem must be solved during translation, for otherwise the abstraction would not be meaningful at all. At the other end of the spectrum, for any language $\mathcal{L}$, an "abstraction" is provided by $\mathcal{L}$ itself. In contrast to the two-element target language $\{yes, no\}$, using $\mathcal{L}$ as target language makes it trivial to translate $\mathcal{L}$ into the abstraction, but the target language does in fact not abstract at all and devising a semantics for the abstraction is as hard as devising a semantics for the original language.

Thus abstraction proper should indeed disregard some information, but not too much of it. In the example above, the fact that the language $\{yes, no\}$ can abstract away from any decision problem is no argument for its usefulness as an abstraction formalism, since its expressive power is clearly too poor to model real problems (meaning problems that are syntactically different from their solutions). Consequently the expressiveness of a language is important when using it as a target language for abstraction. More specifically, a suitable target language for abstraction must be expressive enough to model important problem aspects, while being sufficiently abstract to ignore irrelevant details.

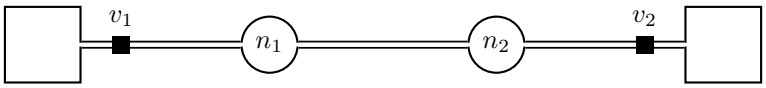
So to be able to use a formalism for abstraction, we obviously need a clear picture of its capabilities as a KR language, especially its expressive power in comparison to other languages, and about the properties of its semantics. It is the main objective of this paper to provide this

information for abstract dialectical frameworks. For this purpose, we technically view ADFs as KR languages – but of course our work has ramifications for ADFs as abstraction formalisms. In the same way as there is no single intended semantics for argumentation frameworks, there is also no single perfect formalism for abstraction. But to be able to make an informed choice, it is of great importance to understand the inherent relationships between different available options. Our results will facilitate this choice and be an aid to anyone wishing to abstract from concrete argumentation languages; especially, our results will help them decide if they want to translate into AFs or into ADFs.

But why, after all, should there be a choice to be made between AFs and ADFs? Here, the additional expressiveness of ADFs in comparison to AFs comes into play. As we will see throughout the paper, the well-known distinction between supported and stable models from logic programming is present in ADFs but is missing in AFs. In a different disguise, this same distinction also materialises as Moore expansions vs. Reiter extensions in nonmonotonic logics [12]. To summarise it in a nutshell, there are basically two ways in which the major nonmonotonic KR formalisms deal with cyclic positive dependencies between pieces of knowledge. To explain what such cyclic support dependencies are and why they can be problematic, let us look at a study from the literature where researchers applied several logic-based knowledge representation techniques in a medium-sized practical application.

Nogueira et al. [32] describe a declarative rule-based system that controls some of the functions of a space shuttle. More specifically, the system operates the space shuttle's reaction control system, whose primary responsibility is to manoeuvre the shuttle through space. Part of the rule-based specification represents the plumbing system of this reaction control system. The plumbing system consists of a collection of tanks, jets and pipe junctions, which are connected through pipes. The flow of fluids through pipes is controlled by valves. The purpose of the plumbing system is to deliver fuel and oxidiser from tanks to the jets needed to perform a manoeuvre. The structure of the plumbing system is described by a directed graph whose nodes are tanks, jets and pipe junctions, and whose edges are labelled by valves. The description of the plumbing system should predict how the positions of valves affect the pressure of tanks, jets and junctions. For tanks themselves, the pressure resulting from pressurising certain (other) tanks is easy to specify. For all other nodes in the graph the definition is recursive: roughly, any non-tank node is pressurised by a tank if the node is connected by an open valve to a node which is pressurised by the tank. Nogueira et al. [32] explicitly recognise that modelling this is non-trivial because the connection graph of the plumbing system can contain cycles. That is, there may be nodes in the graph that are mutually connected to each other, and accurately modelling this is not straightforward:

Example 1.1 (Under Pressure)**.** Consider the following easy setup where two nodes n_1, n_2 with associated tanks are connected to each other. The connection between a node n_i and its tank is controlled by the valve v_i in between.



For the purpose of this example, we assume that the tanks are pressurised. Then obviously, opening v_1 pressurises n_1; likewise, opening v_2 pressurises n_2. But due to the connection in between, it is also the case that pressurising n_1 indirectly pressurises n_2, and pressurising n_2 indirectly pressurises n_1. The easiest way to express all of this in logic programming is via the

four rules

$$n_1 \leftarrow v_1 \qquad n_2 \leftarrow v_2$$
$$n_1 \leftarrow n_2 \qquad n_2 \leftarrow n_1$$

where the atoms n_1, n_2 express that the respective node is pressurised, and v_1, v_2 express that the respective valve is open. This way of representing the domain is very elegant in that it is modular: specifying additional parts of the system can be easily achieved by adding new rules – previous rules need not be modified. This is especially important since the real system is going to be considerably more complex.

Now the Clark completion [10] of this program is given by the four propositional formulas $n_1 \equiv (v_1 \vee n_2)$, $n_2 \equiv (v_2 \vee n_1)$, $v_1 \equiv \bot$ and $v_2 \equiv \bot$. So the valves are considered not open because there are no rules with head v_1 or v_2. The common models of the formulas in the Clark completion lead to the supported model semantics of this program, which considers two states to be possible: $\emptyset$ (where neither of the nodes is pressurised) and $\{n_1, n_2\}$ (where both nodes are pressurised).

But of course, causality dictates that the two nodes cannot simply pressurise each other without an external cause (that is, through an open valve). A reasoner that predicts "both nodes are pressurised" as possible successor state of the state "both nodes are not pressurised" when no relevant valve has been opened in between is obviously not of great assistance – only more so if it offers the cyclic explanation "one node is pressurised because the other is." So the knowledge engineers that specify and use the system should be aware that the supported model semantics does not accurately reflect causality in this domain.

On the other hand, the set $\emptyset$ is the only stable model of the logic program, showing that the stable model semantics correctly deals with the issue at hand. And indeed, Nogueira et al. [32] explicitly remarked that the ability of answer set programming to express and to reason with recursion allowed them to use a concise definition of pressure.

Such issues with cyclic support dependencies not only occur in logic programs, but also in default logic and autoepistemic logic:

- Cyclic support is allowed by supported semantics for logic programs (which is equivalent to the Clark completion [10]) and in expansions of autoepistemic logic [29].
- Cyclic support is disallowed by stable semantics for logic programs [20] and in extensions of default logic [35].[1]

The fact that this distinction is not present in AFs means that anyone translating their modelling language into AFs has to take care of the issue of cyclic support themselves and thus has to solve part of the problem by hardwiring it into the translation. (Just like a decision problem has to be solved when it is "translated" into the language $\{yes, no\}$.) When ADFs are used as a target language, ADF semantics will simply take care of cyclic supports, thereby considerably simplifying the translation.

Generally speaking, it is at the heart of an abstraction to remove information; it is at the heart of a *good* abstraction to remove *irrelevant* information. If some removed information afterwards turns out to have been relevant, it either has to be (however costly) recomputed or is simply lost. And if the target language cannot natively express some concept, then information about this concept is bound to get lost. An example, again, is the support relation between atoms in a

[1] But this is not inherent to these formalisms – both strong expansions for autoepistemic logic that reject cyclic support, and weak extensions for default logic that accept cyclic support can be defined [12].

logic program, which is hardly translated into an AF and easily translated into an ADF as this paper will show.

More concrete empirical evidence for the usefulness of abstract dialectical frameworks has already been provided in the literature. For one, Brewka and Gordon [2] translated argument evaluation structures of the Carneades framework [21] into ADFs.[2] It is especially remarkable that their work allowed cyclic dependencies among arguments, which was previously not possible in Carneades. Meanwhile, Van Gijzel and Prakken [40] also translated Carneades into AFs via ASPIC+ [33]. They can deal with cycles, but even with cycles there is only one unique stable, preferred, complete, grounded extension. Thus the semantic richness of abstract argumentation is not used, and more importantly the user cannot choose whether they want to accept or reject positive cyclic dependencies between arguments. In contrast, in the ADF approach of Brewka and Gordon [2], the user can choose whether support cycles should be accepted or rejected, by choosing models or *stable* models as intended ADF semantics. For another, we [37] have shown how ADFs can be used to provide an argumentation-based semantics for the defeasible theories of Caminada and Amgoud [8]. Our translated ADFs treat the problematic examples from the literature [8, 44] in the right way, and we proved that the translated frameworks satisfy the rationality postulates proposed by Caminada and Amgoud [8]. The translation is efficiently computable, since it involves only a quadratic blowup and creates a number of arguments that is linear in the size of the defeasible theory. Furthermore, the frameworks can detect cyclic supports amongst literals via ADFs' stable semantics.

To summarise, our main arguments for using abstract dialectical frameworks as abstraction language are the following conclusions of this paper:

- ADFs are at least as expressive as AFs, and thus can represent all important problem aspects that AFs can represent. On top of that, ADFs offer a built-in treatment of positive cyclic dependencies which is derived from decades of research into nonmonotonic knowledge representation languages.

- ADFs are at most as expressive as normal logic programs, and therefore still sufficiently simple to be suited as an abstraction formalism.

- ADFs provide all of Dung's standard semantics for AFs, so there is no loss in semantical richness. On the contrary, each of the standard AF semantics (stable, preferred, complete, grounded) has at least *two* ADF generalisations.

To go about our main task of analysing the expressiveness of abstract dialectical frameworks, we do not have to start from scratch. Brewka and Woltran [3] already showed that ADFs are at least as general as AFs and also provided a (non-modular) translation from normal logic programs to ADFs that preserves stable models. 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. That translation is technically remarkable since it works irrespective of the specific chosen representation of acceptance conditions, provided the chosen representation is reasonable in a complexity-theoretic sense. However, the translation depends on the particular ADF semantics that is used: 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).

[2] Note that in their approach, an ADF statement corresponds to an argument evaluation structure of Carneades and is hence on the same abstraction level.

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) [11, 12]. 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 [11]. 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 [22] and Janhunen [25] 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 conclusively 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 [12]. 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 naive, stage, preferred and semi-stable semantics. As a corollary and another new contribution, this also defines these semantics for default logic and autoepistemic logic [12]. This is a considerable improvement upon a result by Dung [14], 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 [36], 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 [14], 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 [11, 13], 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 $O(x) \sqsubseteq x$ and a *postfixpoint* of O if $x \sqsubseteq O(x)$. 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 [38]. In particular, its least fixpoint $lfp(O)$ exists; additionally, the least prefixpoint of O is also its least fixpoint.

2.1. The Algebraic Framework of Denecker et al. [11]

Building upon the fundamental result by Tarski and Knaster, Denecker et al. [11] introduce the important concept of an approximation of an operator. In the study of semantics of non-monotonic 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* $\mathcal{O}$.[3] When O operates on a set L, its approximation $\mathcal{O}$ operates on pairs $(x, y) \in L^2$ where L^2

[3]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.[4]

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. [11] 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. [11] 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 $c\mathcal{O}(y) \stackrel{\text{def}}{=} lfp(\mathcal{O}'(\cdot, y))$;
- *stable operator for* $\mathcal{O}$ as $\mathcal{SO} : L^2 \to L^2$ by $\mathcal{SO}(x, y) \stackrel{\text{def}}{=} (c\mathcal{O}(y), c\mathcal{O}(x))$.

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 [12].

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 $\mathcal{O}$:

[4] Denecker et al. [11] call such pairs "complete," we however use that term for argumentation in a different meaning and want to avoid confusion.

Kripke-Kleene semantics	$lfp(\mathcal{O})$
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	$lfp(\mathcal{SO})$
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 $(x, y) = \mathcal{SO}(x, y) = (c\mathcal{O}(y), c\mathcal{O}(x)) = (lfp(\mathcal{O}'(\cdot, y)), lfp(\mathcal{O}'(\cdot, x)))$ 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. [13] 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 $\mathcal{U}_O(x, y) \stackrel{\text{def}}{=} (\mathcal{U}'_O(x, y), \mathcal{U}''_O(x, y))$ with

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

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 [18], whose fixpoint-theoretic approach to logic programming was extended by Denecker et al. [11]. For a nonempty set A – the *signature*, or set of *atoms* –, define $not\, A \stackrel{\text{def}}{=} \{not\, a \mid a \in A\}$ and the set of *literals* over A as $Lit(A) \stackrel{\text{def}}{=} A \cup not\, A$. A *logic program rule* over A is then of the form $a \leftarrow M$ where $a \in A$ and $M \subseteq Lit(A)$. The rule can be read as logical consequence, "a is true if all literals in M are true." We denote by $M^+ \stackrel{\text{def}}{=} M \cap A$ and $M^- \stackrel{\text{def}}{=} \{a \in A \mid not\, a \in M\}$ 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)* Π 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_Π associated with a definite logic program Π [18]. 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 Π. 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 [18] 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 Π over A, define an (approximating) operator $\mathcal{T}_\Pi : 2^A \times 2^A \to 2^A \times 2^A$ as follows: for $X, Y \subseteq A$,

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

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 Π; its consistent fixpoints are the three-valued supported models of Π. The two-valued supported models of Π are computed by the abovementioned operator T_Π, that – in this setting – is defined by $T_\Pi(M) = \mathcal{T}'_\Pi(M,M)$ [11].

The abstract principles of Denecker et al. [11] 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 Π. The stable operator $\mathcal{ST}_\Pi$ also gives rise to the *well-founded model* of Π, 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" [36]; *L-stable models* are three-valued stable models (X,Y) where $Y \setminus X$ is $\subseteq$-minimal – L-stable is for "least undefined" [36]. 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_{π_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 π_1. Now we add another rule to this program and set $\pi_2 \stackrel{\text{def}}{=} \pi_1 \cup \{c \leftarrow \{b, not\, d\}\}$, 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 π_2.

2.3. Abstract Argumentation Frameworks

Dung [14] introduced a way to study the fundamental mechanisms that humans use in argumentation. His argumentation frameworks (AFs) Θ are pairs (A,R) where A is a set and $R \subseteq A \times A$. The intended reading of an AF Θ 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 $\mathit{Attackers}_\Theta(a) \stackrel{\text{def}}{=} \{b \in A \mid (b, a) \in R\}$. 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 $\mathit{Attacked}_\Theta(S) \stackrel{\text{def}}{=} \{b \in A \mid (a, b) \in R \text{ for some } a \in S\}$. 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 [14] already studied. The first is the *characteristic function* of an AF $\Theta = (A, R)$: for $S \subseteq A$, define $F_\Theta(S) \stackrel{\text{def}}{=} \{a \in A \mid S \text{ defends } a\}$. This operator F_Θ is $\subseteq$-monotone and therefore has a least fixpoint in the lattice $(2^A, \subseteq)$. This least fixpoint of F_Θ is defined as the *grounded* extension of Θ. The second relevant operator U_Θ 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") – formally $U_\Theta(S) \stackrel{\text{def}}{=} A \setminus \mathit{Attacked}_\Theta(S)$. It is an antimonotone operator, and its fixpoints are the *stable* extensions of Θ. Additionally, U_Θ 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_Θ. More generally, a set $S \subseteq A$ is *admissible* iff S is conflict-free and $S \subseteq F_\Theta(S)$. Finally, *preferred* extensions are $\subseteq$-maximal complete extensions; and *semi-stable* extensions are those complete extensions E where the set $E \cup \mathit{Attacked}_\Theta(E)$ (the *range* of the extension E) is $\subseteq$-maximal. The same maximisation criteria that lead from admissible sets to preferred and semi-stable extensions can also be applied to conflict-free sets: a *naive extension* of an AF is a $\subseteq$-maximal conflict-free set; a *stage extension* of an AF is a conflict-free set with $\subseteq$-maximal range. For two argumentation frameworks $\Theta_1 = (A_1, R_1)$ and $\Theta_2 = (A_2, R_2)$, their union is defined as $\Theta_1 \cup \Theta_2 \stackrel{\text{def}}{=} (A_1 \cup A_2, R_1 \cup R_2)$.

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 θ.

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 $\mathit{par}(s) \stackrel{\text{def}}{=} \{r \in S \mid (r, s) \in L\}$
- $C = \{C_s\}_{s \in S}$ 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 $par(s)$. Alternatively, any such function C_s can be represented by the set of all parent subsets leading to acceptance, $C_s^{in} \stackrel{\text{def}}{=} \{M \subseteq par(s) \mid C_s(M) = in\}$. We will use both representations in this paper and indicate the alternative one by writing an ADF as (S, L, C^{in}).

Many more specific representations of acceptance conditions are possible, Brewka and Woltran [3] even introduce two of these additional representations: For one, an acceptance condition C_a can be described via a propositional formula φ_a over the vocabulary $par(a)$, which is straightforward to use whenever each statement has only finitely many relevant parents. The understanding there is that C_a^{in} is given by the two-valued models of φ_a, where an interpretation is identified with the set of atoms that are evaluated to true. For another, Brewka and Woltran [3] also demonstrated how assigning weights to links and combining these weights with proof standards can give rise to acceptance conditions.

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* Ξ 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 $M_1 = \{a, b, c\}$ and $M_2 = \{a, d\}$. Intuitively, a is always in and thus contained in both models. For the self-supporting b, the model semantics has a choice whether or not to accept it, and this choice determines the two models. In M_1, statement b is accepted along with a, their joint support of c becomes relevant and c is also accepted. (Statement d is not accepted by M_1 since b is accepted and attacks d.) In M_2, statement b is not accepted whence c is not accepted due to a lack of support; statement d behaves like an AF argument and so is accepted because its only attacker b is not accepted.

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* Ξ 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* Ξ 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 supporting or attacking; we use L^+ to denote all supporting and L^- to denote all attacking links of L in Ξ. A model M of a bipolar ADF Ξ is a *BW-stable model of* Ξ iff it is the least model of the reduced ADF Ξ^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* Ξ 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* Ξ iff it is $\subseteq$-maximal among the sets BW-admissible in Ξ. 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[5]

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

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

Example 2.1 (Continued)**.** The $\leq_i$-least fixpoint of Γ_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

[5] 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 Γ''_Ξ for computing the second component.

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 Γ_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 $C^{in}_a = \{\emptyset\}$ 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 Θ and the BW-well-founded model of $\Xi(\Theta)$;*
2. *conflict-free sets of Θ and conflict-free sets of $\Xi(\Theta)$;*
3. *stable extensions of Θ and models of $\Xi(\Theta)$;*
4. *stable extensions of Θ and BW-stable models of $\Xi(\Theta)$;*
5. *preferred extensions of Θ 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 for which we will provide an intuitive explanation.

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 [18]. 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 from the two-valued $\{in, out\}$ 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 Ξ, 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

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

The last condition $(par(s) \setminus B) \cap Y = \emptyset$ can be equivalently reformulated as $par(s) \setminus B \subseteq S \setminus Y$. By $B \subseteq X$ this means that all parents of s which are not *in* must be *out* – there must not be undecided parents of s.

A two-valued immediate consequence operator for ADFs (the equivalent of logic programs' two-valued van Emden-Kowalski operator T_Π) is now given by $G_\Xi(X) \stackrel{\text{def}}{=} \mathcal{G}'_\Xi(X,X)$. 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.

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

□

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 $M \in C^{in}_s$ 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 [38]. From $\mathcal{G}_\Xi$ being approximating it follows that it maps consistent pairs to consistent pairs [11, 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. [11]. 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 M is conflict-free for an ADF Ξ iff application of the two-valued immediate consequence operator G_Ξ leads to a superset of M, that is, M is a postfixpoint of G_Ξ. Intuitively speaking, each statement that is contained in a conflict-free set M has a reason to be contained in M.

Proposition 3.3. *For any abstract dialectical framework $\Xi = (S, L, C)$, a set $M \subseteq S$ is conflict-free for Ξ iff $M \subseteq G_\Xi(M)$.*

Proof.

$$
\begin{aligned}
&M \text{ is conflict-free} \\
&\text{iff for all } s \in M \text{ we have } M \cap \mathit{par}(s) \in C_s^{in} \\
&\text{iff } M \subseteq \left\{ s \in S \;\middle|\; M \cap \mathit{par}(s) \in C_s^{in} \right\} \\
&\text{iff } M \subseteq G_\Xi(M) \qquad \text{(by Lemma 3.1)} \quad \square
\end{aligned}
$$

Notice that this characterisation only uses conflict-free *sets* and is thus inherently two-valued. We will later generalise "conflict-free" to three-valued interpretations represented by consistent 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 Ξ iff $\mathcal{G}_\Xi(M, M) = (M, M)$.*

Proof.

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

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 ξ: The reduct ξ^M is given by the triple $(\{b\}, \emptyset, \{C_b'^{in}\})$ with $C_b'^{in} = \{\emptyset\}$, an ADF where b is always *in*. (The link (b, b) is not in the reduct because it is attacking in ξ.) 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: ξ 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 $\mathcal{G}'_\Xi(\cdot, M)$ and G_{Ξ^M} whenever M is a model of Ξ.

Lemma 3.5. *Let $\Xi = (S, L, C)$ be a bipolar ADF and (M, M) be a two-valued supported model for Ξ. For any $X \subseteq M$ we find $\mathcal{G}'_\Xi(X, M) \subseteq G_{\Xi^M}(X)$.*

Proof. Recall that the reduct of Ξ with M is defined by $\Xi^M = (M, L^M, C^M)$ with reduced links $L^M = \{(r, s) \mid r, s \in M, (r, s) \in L^+\}$ and for each $s \in M$ and $B \subseteq M$, we have $C^M_s(B) = in$ iff $C_s(B) = in$. 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^M_s the parent nodes of s with respect to L^M. It follows that $P^M_s = (M \cap P_s) \setminus \{r \in P_s \mid (r, s) \notin L^+\}$.

Let $s \in \mathcal{G}'_\Xi(X, M)$. (Observe that $X \subseteq M$ means $\mathcal{G}'_\Xi(X, M) \subseteq \mathcal{G}'_\Xi(M, M) = M$ and thus $s \in M$.) Then there is a $B \subseteq P_s$ with $C_s(B) = in$, $B \subseteq X$ and $(P_s \setminus B) \cap M = \emptyset$. Now $P^M_s \subseteq P_s$ and $X \subseteq M$ yield $(P^M_s \setminus B) \cap X = \emptyset$, whence $X \cap P^M_s \subseteq B$. Define $B' = B \setminus \{r \in P_s \mid (r, s) \notin L^+\}$. By definition $B' \subseteq P^M_s$, whence by $B' \subseteq B \subseteq X$ we get $B' \subseteq X \cap P^M_s$. Since all the removed parents r were attackers (Ξ is bipolar), we still have $C_s(B') = in$. Now all links from P^M_s to s are supporting and thus still $C_s(X \cap P^M_s) = in$. Hence $C_s(X \cap P^M_s) = C^M_s(X \cap P^M_s) = in$ and $s \in G_{\Xi^M}(X)$. □

This shows that G_{Ξ^M} – the two-valued operator associated to the reduced ADF Ξ^M – is in some sense "complete" with respect to the result of $\mathcal{G}'_\Xi(\cdot, M)$ – the operator for checking whether M is a two-valued stable model of Ξ. The next lemma will show that this "completeness" carries over to the least fixpoints of these operators.

Lemma 3.6. *Let $\Xi = (S, L, C)$ be a bipolar ADF and (M, M) be a two-valued supported model for Ξ. If M is the least fixpoint of $\mathcal{G}'_\Xi(\cdot, M)$, then it is the least fixpoint of G_{Ξ^M}.*

Proof. We use the notation from the proof of Lemma 3.5. Let $s \in M$ and observe that we have $C_s(M \cap P_s) = in$ since M is a model of Ξ. By the definition of the reduct, we get $P^M_s = (M \cap P_s) \setminus \{r \in P_s \mid (r, s) \notin L^+\}$. Since Ξ is bipolar, any link from $(M \cap P_s) \setminus P^M_s$ is attacking and thus $C_s(P^M_s) = in$.

- M is a fixpoint of G_{Ξ^M}:

$$
\begin{aligned}
& G_{\Xi^M}(M) \\
&= \left\{ s \in M \;\middle|\; C^M_s(M \cap P^M_s) = in \right\} && \text{(by definition of } G_{\Xi^M}) \\
&= \left\{ s \in M \;\middle|\; C^M_s(P^M_s) = in \right\} && (P^M_s \subseteq M) \\
&= \left\{ s \in M \;\middle|\; C_s(P^M_s) = in \right\} && \text{(by definition of } C^M_s) \\
&= \{ s \in M \mid C_s(M \cap P_s) = in \} && \text{(see above)} \\
&= \{ s \in S \mid C_s(M \cap P_s) = in \} && (s \in S \setminus M \text{ iff } C_s(M \cap P_s) = out) \\
&= G_\Xi(M) && \text{(By definition of } G_\Xi) \\
&= M && (M \text{ is a model of } \Xi)
\end{aligned}
$$

- M is the *least* fixpoint of G_{Ξ^M}: Let $X \subseteq M$ be a fixpoint of G_{Ξ^M}. By Lemma 3.5, $\mathcal{G}'_\Xi(X, M) \subseteq G_{\Xi^M}(X) = X$ and X is a prefixpoint of $\mathcal{G}'_\Xi(\cdot, M)$. Since M is the least fixpoint and thus also the least prefixpoint of $\mathcal{G}'_\Xi(\cdot, M)$, we get $M \subseteq X$. □

Using the lemma, it is easy to show that the set of BW-stable models contains all operator-based two-valued stable models.

Proposition 3.7. *Let $\Xi = (S, L, C)$ bipolar abstract dialectical framework and $M \subseteq S$. If (M, M) is a two-valued stable model of Ξ, then M is a BW-stable model of Ξ.*

Proof. Let $\mathcal{SG}_\Xi(M, M) = (M, M)$. By definition $M = c\mathcal{G}_\Xi(M) = lfp(\mathcal{G}'_\Xi(\cdot, M))$, that is, M is the least fixpoint of $\mathcal{G}'_\Xi(\cdot, M)$. By Lemma 3.6, M is the least fixpoint of G_{Ξ^M}. Therefore, M is the least model of Ξ^M (and a model of Ξ), thus it is a BW-stable model of Ξ. □

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. [13], 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 ξ has two BW-stable models, $M_1 = \{a\}$ and $M_2 = \{a, b\}$: The reduct of ξ 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 BW-stable model $M_2 = \{a, b\}$, the reduct of ξ 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 in fact irrelevant.) 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 ξ admits only the single two-valued stable model $(\{a\}, \{a\})$.

The problem with this example is that the ADF ξ allows for the BW-stable model M_2 in which statement b cyclicly supports itself. This violates the intuitive requirement of stable semantics that whatever it takes to be true must have a non-cyclic justification. Furthermore, in logic programming, two distinct stable models of normal logic programs 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 3.8. *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. However, 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 Ξ 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 3.9. *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 Ξ.*

Proof. First observe that we find the two-valued consequence operator of the reduct Ξ_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_{Ξ_M} and $\mathcal{G}'_\Xi(\cdot, M)$ coincide on 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_{Ξ_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, that is, $\xi = (S, L, C)$ 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 ξ with $M_2 = \{a, b\}$ is given by $\xi_{M_2} = (S, L, C_{M_2})$ with $C_{M_2,a}^{in} = \{\emptyset\}$ and $C_{M_2,b}^{in} = \{\{a, b\}\}$. It is easy to see that $\{a\} \neq M_2$ is the least model of this ADF and M_2 is not a stable model of ξ.

The (new) reduct of ξ with $M_1 = \{a\}$ is given by $\xi_{M_1} = (S, L, C^{in}_{M_1})$ with $C^{in}_{M_1,a} = \{\emptyset\}$ and $C^{in}_{M_1,b} = \{\{a, b\}\}$. Its least model is $\{a\} = M_1$ and M_1 is thus a stable model of ξ, 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 abstract dialectical framework with a self-supporting cycle between two arguments, $\xi = (S, L, C)$ with $S = \{a, b\}$, $L = \{(a, b), (b, a)\}$ and $C^{in}_a = \{\{b\}\}$, $C^{in}_b = \{\{a\}\}$. In other words, the links between a and b are both supporting. Hence the set $\{a, b\}$ is a (two-valued supported) model of ξ, but it is not BW-admissible: $\{a, b\}$ is not a stable model of ξ or any subframework of ξ.

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 $C^{in}_a = \{\emptyset, \{b\}, \{c\}\}$, $C^{in}_b = \{\emptyset\}$, $C^{in}_c = \{\{d\}\}$ and $C^{in}_d = \{\emptyset\}$. 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 ξ 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 ξ' has the stable model $\{a, b\}$, which is easily verified when looking at the reduct $\xi'^M = (\{a, b\}, \emptyset, C'^M)$ where $C'^M_a = C'^M_b = \{\emptyset\}$. 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.[6]

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.

Intuitively, we require that an admissible pair is first of all consistent and satisfies a further criterion: for any statement that is labelled with either *in* or *out*, the pair must provide sufficient

[6] 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].

justification for this choice. For a pair (M, N), this means that any statement that is labelled *in* (contained in M) must indeed be accepted by this pair; conversely, any statement that is labelled *out* (not contained in N) must indeed be rejected by the pair. Acceptance and rejection is expressed using the approximating operator, so for (M, N) we require $M \subseteq \mathcal{G}'_\Xi(M, N)$ (justified lower bound) and $\mathcal{G}''_\Xi(M, N) \subseteq N$ (justified upper bound). This combination is easily expressed using the information ordering.

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

It is clear that the lower bound of an admissible pair (M, N) is a conflict-free set since $M \subseteq \mathcal{G}'_\Xi(M, N) \subseteq \mathcal{G}'_\Xi(M, M) = G_\Xi(M)$. 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. Interestingly, $\leq_i$-postfixpoints of operators $\mathcal{O}$ were also important for Denecker et al. [13] – they called them $\mathcal{O}$*-reliable* pairs.

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 3.10. *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, $(x, y) \leq_i \mathcal{O}(x, y)$ and there is no admissible pair $(\hat{x}, \hat{y})$ with $(x, y) <_i (\hat{x}, \hat{y})$. We have to show $\mathcal{O}(x, y) = (x, y)$, so assume to the contrary that $\mathcal{O}(x, y) \not\leq_i (x, y)$, that is, $(x, y) <_i \mathcal{O}(x, y)$. Since $\mathcal{O}$ is approximating, it is in particular $\leq_i$-monotone and from $(x, y) \leq_i \mathcal{O}(x, y)$ we can infer $\mathcal{O}(x, y) \leq_i \mathcal{O}(\mathcal{O}(x, y))$. Thus $\mathcal{O}(x, y)$ is itself admissible and $(x, y) <_i \mathcal{O}(x, y)$, in contradiction to (x, y) being $\leq_i$-maximal admissible. □

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 3.11. *Let Ξ 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 Γ_Ξ 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 Γ_Ξ 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 consequence operator G_Ξ.[7]

Lemma 3.12. *For any abstract dialectical framework Ξ, the operator Γ_Ξ is the ultimate approximation of G_Ξ.*

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

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

Now by [13, Theorem 5.6], for $X \subseteq Y \subseteq S$, the ultimate approximation $\mathcal{U}_\Xi$ of the operator G_Ξ 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 $Z \cap \mathit{par}(s) \in C^{in}_s$ iff $s \in G_\Xi(Z)$, which leads to the equalities

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

and, likewise for the upper bound,

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

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 3.13. *For any ADF Ξ, its BW-well-founded semantics coincides with its ultimate Kripke-Kleene semantics.*

[7] According to personal communication this was conjectured by Mirosław Truszczyński.

The well-founded semantics of Ξ 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. (Recall that $M_2 = \{a, d\}$ is the supported model of D where the self-support of b was rejected.)

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 logic program rules for each statement of a given ADF Ξ. The body of a rule 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.

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

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

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

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 ξ from Example 3.1 is translated into the logic program consisting of the following rules:

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

This somewhat obviates why there is no two-valued stable model for ξ: 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, $\overline{B} = par(s) \setminus B$.

Lemma 3.14. *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 $\mathcal{G}'_\Xi(X,Y) = \mathcal{T}'_{\Pi(\Xi)}(X,Y)$.

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

□

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

Theorem 3.15. *Let $\Xi = (S, L, C^{in})$ be an abstract dialectical framework and $\Pi(\Xi)$ its standard logic program. Then Ξ 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 $\Xi_1 = (S_1, L_1, C_1^{in})$ and $\Xi_2 = (S_2, L_2, C_2^{in})$ be ADFs such that $S_1 \cap S_2 = \emptyset$. Then the union of the two ADFs is defined as $\Xi_1 \cup \Xi_2 \stackrel{\text{def}}{=} (S_1 \cup S_2, L_1 \cup L_2, C_1^{in} \cup C_2^{in})$. 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 $\xi_1 = (S_1, L_1, C_1^{in})$ with $S_1 = \{a, b\}$, $L_1 = \{(b, a)\}$, $C_{1,a}^{in} = \{\{b\}\}$ and $C_{1,b}^{in} = \{\emptyset\}$ (in words, b is always *in* and supports a); and $\xi_2 = (S_2, L_2, C_2^{in})$ with $S_2 = \{a, c\}$, $L_2 = \{(c, a)\}$, $C_{2,a}^{in} = \{\{c\}\}$ and $C_{2,c}^{in} = \{\emptyset\}$ (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 $C_{1,a}^{in} \cup C_{2,a}^{in} = \{\{b\}, \{c\}\}$, 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 Π be a normal logic program over a set A of atoms. Define an ADF $\Xi(\Pi) = (A, L, C^{in})$ as follows.

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

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

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

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 $\pi = \{c \leftarrow \emptyset, b \leftarrow not\, b\}$:

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

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, $C_b^{in} = \{\emptyset\}$, statement c being always *in*, $C_c^{in} = \{\emptyset\}$ and statement a being *in* if b is *in*, c is *in*, or both, $C_a^{in} = \{\{b\}, \{c\}, \{b, c\}\}$. However, the original logic programs π_1 and π_2 do not have the same three-valued models: While the only (three-valued supported) model of π_1 is $(\{a, c\}, \{a, b, c\})$, the only (three-valued supported) model of π_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 π), the result is that a is true in π_1 (the disjunction $b \vee c$ evaluates to true) but undefined in π_2 (all the conjuncts $b \wedge \neg c$, $c \wedge \neg b$ and $b \wedge c$ 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_Π for a logic program Π and $G_{\Xi(\Pi)}$ for the logic program's ADF $\Xi(\Pi)$ in the following lemma.

Lemma 3.16. *For any normal logic program Π, we have $T_\Pi = G_{\Xi(\Pi)}$.*

Proof. Abbreviate $\Xi(\Pi) = \Xi$, let A be the signature of Π 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 $B \stackrel{\text{def}}{=} par(a) \cap X$. 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 $B \in C_a^{in}$. □

From the proof we can read off that Π can derive anything that $\Xi(\Pi)$ can derive, for any three-valued pair; in the converse direction, this only works for two-valued pairs. As an immediate consequence, we get correspondence of two-valued supported models.

Corollary 3.17. *Let Π 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 3.16, 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 Π.

Lemma 3.18. *Let Π 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 3.16 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 3.16 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 – that is, it contains no information. 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 ξ.

As much more interesting consequence of Lemma 3.16, it follows that the ultimate approximations of T_Π and $G_{\Xi(\Pi)}$ are identical, thus Π and $\Xi(\Pi)$ also coincide on all ultimate semantics, including ultimate stable models. This means that whatever "goes missing" in the translation from Π 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 [13]. 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 [6] 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 Θ is also a special case of an operator: it is the canonical approximation of U_Θ, 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 $C^{in}_a = \{\emptyset\}$ for each $a \in A$. Thus for any argument $a \in A$, we find that $par(a) = Attackers_\Theta(a)$. Now

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

□

In the remainder, we will denote the four-valued approximation operator of an argumentation framework Θ by $\mathcal{F}_\Theta$; we formally define $\mathcal{F}'_\Theta \stackrel{\text{def}}{=} \mathcal{G}'_{\Xi(\Theta)}$. 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 Θ, 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_Θ (which is at the heart of $\mathcal{F}_\Theta$) can emulate the characteristic function F_Θ of an argumentation framework: F_Θ is the same as twofold application of U_Θ.

Lemma 4.3 ([14, Lemma 45])**.** *For any AF Θ, we have $F_\Theta = U^2_\Theta$.*

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

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

There are several works in the literature that redefine argumentation semantics in terms of (pre-/post-)fixpoints of the two operators F_Θ and U_Θ [1, 23]. 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 Θ 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 Θ iff for some $E' \subseteq A$ the pair (E, E') is a consistent fixpoint of $\mathcal{F}_\Theta$.*

Proof.

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

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 Θ 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 Θ 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 [14, 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 Θ 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 version of "admissible" (Definition 3.3) is a proper generalisation of the respective AF notion. This is easily shown using the 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 Θ 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:

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

□

Jakobovits and Vermeir [24] introduced four-valued labellings for argumentation frameworks. Using indicators for acceptance ($+$) and rejection ($-$), they define the labels $\mathit{in} = \{+\}$, $\mathit{out} = \{-\}$, $\mathit{undec} = \{+, -\}$ and $\mathit{irrelevant} = \emptyset$. It is possible to adapt our intuition behind pairs (X, Y) of sets of arguments in the sense that those in $X \setminus Y$ are irrelevant (instead of inconsistent); in this case, their labels can be seen as indicating which of the two possible statuses $+$ and $-$ are still considered possible for the argument in question. Under this assumption, it is straightforward to adapt the definitions of [24] to our setting. To this end, we first recall their original definition of four-valued labellings.

Definition 4.1 ([24, Definition 3.1])**.** Let $\Theta = (A, R)$ be an AF. A *JV-labelling* is a function $l : A \to 2^{\{+,-\}}$ such that for all $a \in A$:

1. If $- \in l(a)$, then there is a $b \in A$ with $(b, a) \in R$ and $+ \in l(b)$.
2. If $+ \in l(a)$, then $(b, a) \in R$ implies $- \in l(b)$.
3. If $+ \in l(a)$, then $(a, c) \in R$ implies $- \in l(c)$.

A JV-labelling is *total* iff $l(a) \neq \emptyset$ for all $a \in A$.[8]

It is easy to establish a correspondence between pairs (X, Y) and functions $l : A \to 2^{\{+,-\}}$. It only remains to reformulate the three conditions in terms of AF operators, which results in the proposition below.

Proposition 4.10. *Let $\Theta = (A, R)$ be an AF and $X, Y \subseteq A$.*

(A) *The pair (X, Y) corresponds to a JV-labelling iff $X = \mathcal{F}'_\Theta(X, Y)$ and $Y \subseteq \mathcal{F}''_\Theta(X, Y)$.*

(B) *The pair (X, Y) is consistent iff its corresponding JV-labelling is total.*

[8] Jakobovits and Vermeir [24] call such labellings complete, which we however use with a different meaning.

Proof. (A) Define $l : A \to 2^{\{+,-\}}$ such that it maps as follows: $X \cap Y \mapsto \{+\}$, $A \setminus (X \cup Y) \mapsto \{-\}$, $Y \setminus X \mapsto \{+,-\}$ and $X \setminus Y \mapsto \emptyset$. We first observe that for any argument $a \in A$, we have $- \in l(a)$ iff $a \notin X$, and $+ \in l(a)$ iff $a \in Y$. The conditions of Definition 4.1 are now readily formulated thus:

1. If $a \notin X$, then a is attacked by Y; that is, $U_\Theta(Y) \subseteq X$.
2. If $a \in Y$, then $(b, a) \in R$ implies $b \notin X$; that is, $Y \subseteq U_\Theta(X)$.
3. If $a \in Y$, then $(a, c) \in R$ implies $c \notin X$; that is, $X \subseteq U_\Theta(Y)$.

Proposition 4.1 now yields the claim.

(B) (X, Y) is consistent iff $X \subseteq Y$ iff $X \setminus Y = \emptyset$ iff $\{a \in A \mid l(a) = \emptyset\} = \emptyset$ iff l is total. □

Using the truth and information orderings, pairs (X, Y) that correspond to JV-labellings could be characterised by $(X, Y) \leq_t \mathcal{F}_\Theta(X, Y)$ and $\mathcal{F}_\Theta(X, Y) \leq_i (X, Y)$. While this may suggest a possible intuitive reading of such pairs, we have to be cautious as the intuitions underlying these orderings are slightly different. Anyway, when applying the revision operator $\mathcal{F}_\Theta$, the result must be "at least as true" and "at most as informative" as the input pair.

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 pre-/postfixpoints with respect to the information ordering $\leq_i$ and 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 [14] 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 [19, 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 Θ is given by $\Pi(\Xi(\Theta))$, that is, translating the AF Θ 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 3.14 that the associated operators of AFs Θ and their translated logic program $\Pi(\Theta)$ are the same.

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

Now we know from Lemma 4.2 that the approximation operator of any AF Θ 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 Θ, 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. [43, 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 Θ be an AF. The following are identical:*

1. *the grounded extension of Θ, the Kripke-Kleene model of $\Pi(\Theta)$ and the well-founded model of $\Pi(\Theta)$;*
2. *complete extensions of Θ, three-valued supported models of $\Pi(\Theta)$ and three-valued stable models of $\Pi(\Theta)$;*
3. *preferred extensions of Θ, M-supported models of $\Pi(\Theta)$ and M-stable models of $\Pi(\Theta)$;*
4. *semi-stable extensions of Θ, L-supported models of $\Pi(\Theta)$ and L-stable models of $\Pi(\Theta)$;*
5. *stable extensions of Θ, 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 3.14, 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 -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.[9]

[9] Dung's original translation is slightly different; he uses a first-order signature and logic program atoms with variables [14]. Definition 4.2 above is merely a syntactical variant that already incorporates ground instantiation.

Definition 4.2. Let $\Theta = (A, R)$ be an argumentation framework. Define $\text{-}A \stackrel{\text{def}}{=} \{\text{-}a \mid a \in A\}$, $A^{\pm} \stackrel{\text{def}}{=} A \cup \text{-}A$ and a logic program over $A^{\pm}$ as follows.

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

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 4.3. 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 $co(S, P) \stackrel{\text{def}}{=} (S \cup \text{-}\overline{P}, P \cup \text{-}\overline{S})$.[10]

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 Π and Dung translation Π_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 Π_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 $\mathcal{T}'_{\Pi_D}(X, Y) = \{a \mid \text{-}a \notin Y\} \cup$

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

$\text{-}\overline{U_\Theta(X)}$ and $\mathcal{T}'_{\Pi_D}(X, \text{-}\overline{Y}) = Y \cup \text{-}\overline{U_\Theta(X)}$. Now

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

□

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

Proposition 4.15. *Let $\Theta = (A, R)$ be an AF, Π_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 [14, 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 Θ, the Kripke-Kleene model of $\Pi_D(\Theta)$ and the well-founded model of $\Pi_D(\Theta)$;*
2. *complete extensions of Θ, three-valued supported models of $\Pi_D(\Theta)$ and three-valued stable models of $\Pi_D(\Theta)$;*
3. *preferred extensions of Θ, M-supported models of $\Pi_D(\Theta)$ and M-stable models of $\Pi_D(\Theta)$;*
4. *semi-stable extensions of Θ, L-supported models of $\Pi_D(\Theta)$ and L-stable models of $\Pi_D(\Theta)$;*
5. *stable extensions of Θ, 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.

Dung [14] defined two semantics-independent translations from normal logic programs into argumentation frameworks. When restricted to propositional programs, his first translation (Section 4.3.2) is polynomial and faithful with respect to two-valued supported models and the Kripke-Kleene semantics, but not modular. Furthermore it is not faithful with respect to two-valued stable models: the logic program $\{a \leftarrow a\}$ has the only two-valued stable model $\emptyset$, but its associated argumentation framework[11] $(\{a, \neg a\}, \{(a, \neg a), (\neg a, a)\})$ has two stable extensions (corresponding to the logic program's two-valued supported models). For Dung's second translation (Section 4.3.1), the size of the resulting argumentation framework may – in the worst case – be at least exponential in the number of atoms in the vocabulary of the logic program.

Although it is certainly possible to devise polynomial, 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.

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 Ξ 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. 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 consistent pairs that are postfixpoints with respect to the information ordering $\leq_i$. These pairs have the property that applying the revision operator increases (or preserves) their information content. For the sake of completeness we have included the following formal definition.

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 *admissible for* $\mathcal{O}$ iff $(x, y) \leq_i \mathcal{O}(x, y)$.

[11] Actually, applying the translation yields an argumentation framework that looks slightly more complicated: $(\{(\{a\}, a), (\{\neg a\}, \neg a)\}, \{((\{a\}, a), (\{\neg a\}, \neg a)), ((\{\neg a\}, \neg a), (\{a\}, a))\})$. We chose to simplify notation for the sake of readability.

Denecker et al. [13] already took special note of admissible pairs and called them $\mathcal{O}$-reliable. They 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 to all fixpoints of the approximated operator O.

5.2. 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 $x, y \in L$,

- $(x^{-1})^{-1} = x$ ($\cdot^{-1}$ 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 5.2. 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.

5.3. 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.

To generalise this notion to three-valued pairs (X, Y), we require two things:

- any *in* statement (in X) must have a reason not to be *out*, and
- any *out* statement (not in Y) must have a reason to be *out*.

Notice the asymmetry, which resurfaces in the following operator-based definition. For a consistent pair to be conflict-free, we stipulate that applying the approximating operator improves the upper bound of the pair.

Definition 5.3. 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 $x \sqsubseteq \mathcal{O}''(x, y) \sqsubseteq y$.

Let us illustrate this notion on a standard example AF. It shows that conflict-free pairs allow to set arguments to *undec* that attack *in* arguments.

Example 5.1 (Odd Cycle)**.** Consider an attack cycle between three arguments, the AF $\theta = (\{a, b, c\}, \{(a, b), (b, c), (c, a)\})$. Its conflict-free pairs are given by

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

Notice that the single admissible pair $(\emptyset, \{a, b, c\})$ is among the conflict-free pairs.

It is not hard to show that any admissible pair is conflict-free.

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

Proof. The pair (x, y) is admissible if and only if $x \sqsubseteq \mathcal{O}'(x, y)$ and $\mathcal{O}''(x, y) \sqsubseteq y$. Since $\mathcal{O}'(\cdot, y)$ is $\sqsubseteq$-monotone and $\mathcal{O}'(x, \cdot)$ is $\sqsubseteq$-antimonotone, we obtain the following picture:

$$\begin{array}{ccccccccc} x & \sqsubseteq & \mathcal{O}'(x, y) & \leqslant & \mathcal{O}'(x, x) & \leqslant & \mathcal{O}'(y, x) & \sqsubseteq & y \\ & & & \leqslant & \mathcal{O}'(y, y) & \leqslant & & & \end{array}$$

In particular, $x \sqsubseteq \mathcal{O}''(x, y) = \mathcal{O}'(y, x) \sqsubseteq y$ and (x, y) is conflict-free. □

Let us again take note of the asymmetry in the definition of conflict-free pairs. If admissible pairs ensure that during revision, both lower and upper bounds are improved, why should conflict-free pairs be defined such that only the upper bound must improve? Why not improve the lower bound? Another possibility to define conflict-free pairs would have been to say a pair (x, y) is conflict-free for an operator $\mathcal{O}$ iff $x \sqsubseteq \mathcal{O}'(x, y) \sqsubseteq y$. In AF terms, this alternative notion allows to set arguments to *undec* that are attacked by *in* arguments.

Example 5.1 (Continued)**.** In the odd attack cycle between three arguments, the alternative conflict-free pairs (improving the lower bound) are

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

Unfortunately, this possible alternative version of conflict-freeness is not a suitable generalisation of the set-based AF notion, since there are argumentation frameworks with conflict-free sets that are not the lower bound of any such pair.

Example 5.2. Consider the argumentation framework $\theta = (\{a, b\}, \{(a, b)\})$ where a attacks b. The set $X = \{b\}$ is conflict-free. Assume to the contrary that there is a $Y \subseteq A$ such that $X \subseteq U_\theta(Y) \subseteq Y$. It follows that $b \in U_\theta(Y)$, that is, Y does not attack b. Thus Y does not contain a and $Y \subseteq \{b\} = X$. From $X \subseteq Y$ we conclude $X = Y = \{b\}$. But we find that $U_\theta(Y) = \{a, b\} \not\subseteq Y$. Contradiction.

We want to stress that these two different generalisations of conflict-free sets are not an artefact of using approximating operators. Rather, they occur in the step from two-valued to three-valued semantics. As opposed to the alternative, the version requiring improvement of the upper bound generalises conflict-free sets.

Proposition 5.2. *Let $\Theta = (A, R)$ be an AF and $X \subseteq A$. X is conflict-free iff $(X, U_\Theta(X))$ is a conflict-free pair.*

Proof.

$$\begin{aligned}
&X \text{ is conflict-free}\\
\text{iff } &X \subseteq U_\Theta(X)\\
\text{iff } &(X, U_\Theta(X)) \text{ is consistent and } X \subseteq \mathcal{F}''_\Theta(X, Y) \subseteq U_\Theta(X)\\
\text{iff } &(X, U_\Theta(X)) \text{ is a conflict-free pair}
\end{aligned}$$

□

Indeed, this notion of conflict-free pairs for AFs coincides with Caminada's definition of conflict-free labellings [7].

Proposition 5.3. *For any AF $\Theta = (A, R)$ and $X \subseteq Y \subseteq A$, the pair (X, Y) is conflict-free iff the labelling $l : A \to \{in, out, undec\}$ with $X \mapsto in$, $Y \setminus X \mapsto undec$, $A \setminus Y \mapsto out$ is conflict-free in the sense of [7, Definition 3].*

5.4. Naive

It is clear that conflict-free pairs are amenable to the usual maximisation criteria that lead from admissible to preferred and semi-stable semantics. We begin with naive semantics, which for AFs are just $\subseteq$-maximal conflict-free sets.

Definition 5.4. 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 an *M-conflict-free pair for $\mathcal{O}$* iff it is $\leq_i$-maximal among the conflict-free pairs for $\mathcal{O}$.

We keep our uniform naming conventions for recording maximisation criteria. It is straightforward that this definition generalises naive semantics.

Proposition 5.4. *Let $\Theta = (A, R)$ be an AF. A set $X \subseteq A$ is a naive extension of Θ iff the pair $(X, U_\Theta(X))$ is an M-conflict-free pair of Θ.*

Proof. First note that for any $X \subseteq A$, the set $U_\Theta(X)$ is fixed. Furthermore U_Θ is a $\subseteq$-antimonotone operator and thus $X \subseteq Y$ iff $(X, U_\Theta(X)) \leq_i (Y, U_\Theta(Y))$. Consequently for maximisation there is no difference whether we $\subseteq$-maximise the set X or we $\leq_i$-maximise the pair $(X, U_\Theta(X))$. It follows that

$$\begin{aligned}
&X \text{ is a naive extension for } \Theta\\
\text{iff } &X \text{ is conflict-free and } X \text{ is } \subseteq\text{-maximal}\\
\text{iff } &X \subseteq U_\Theta(X) \text{ and } X \text{ is } \subseteq\text{-maximal}\\
\text{iff } &X \subseteq U_\Theta(X) = \mathcal{F}''_\Theta(X, U_\Theta(X)) \text{ and } (X, U_\Theta(X)) \text{ is } \leq_i\text{-maximal}\\
\text{iff } &(X, U_\Theta(X)) \text{ is an M-conflict-free pair of } \mathcal{F}_\Theta
\end{aligned}$$

□

5.5. Stage

Verheij [41] defined the notion argumentation stage for an argumentation framework, in turn based on three-valued status assignments. Such an assignment represents not only the arguments that are accepted (as extensions do), but also those which are not accepted. (Hence each assignment gives rise to a unique extension, but not necessarily vice versa.) An *argumentation stage* is a three-valued status assignment to arguments with the restriction that the arguments that are not accepted must be exactly the ones that have an attacker that is accepted. It follows that

the set of arguments which are accepted in an argumentation stage (its associated extension) is conflict-free. It is easy to find out how (argumentation) stages can be captured in our setting:

Proposition 5.5. *Let $\Theta = (A, R)$ be an argumentation framework. A consistent pair (X, Y) is an argumentation stage iff $Y = U_\Theta(X)$.*

Proof. (X, Y) is a stage iff $A \setminus Y = \mathit{Attacked}_\Theta(X)$ iff $Y = U_\Theta(X)$. □

While an argumentation stage is always a conflict-free pair, and any conflict-free set gives rise to an argumentation stage (and thus to a conflict-free pair), there are conflict-free pairs which are not argumentation stages. (Consider Example 5.2 and the pair $(\{a\}, \{a, b\})$.) Still, we can apply the range maximisation criterion to conflict-free pairs in order to generalise stage extension semantics.

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 an *L-conflict-free pair for $\mathcal{O}$* iff $y \sqcap x^{-1}$ is $\sqsubseteq$-minimal among the conflict-free pairs for $\mathcal{O}$.

For AFs, L-conflict-free pairs coincide with stage extensions.

Proposition 5.6. *Let $\Theta = (A, R)$ be an AF. A set $X \subseteq A$ is a stage extension of Θ iff the pair $(X, U_\Theta(X))$ is an L-stage pair of $\mathcal{F}_\Theta$.*

Proof.

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

□

When trying to define two-valued conflict-free pairs we end up with pairs (x, x) where $x \sqsubseteq \mathcal{O}''(x, x) = \mathcal{O}'(x, x) \sqsubseteq x$, that is, two-valued supported models.

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 σ is seen as a function assigning to a knowledge base κ a set of models, then a partial order on semantics is given by $\sigma_1 \leq \sigma_2$ iff $\sigma_1(\kappa) \subseteq \sigma_2(\kappa)$ for all κ. In the figure, an arrow from σ_1 to σ_2 expresses $\sigma_1 \leq \sigma_2$ – in words, all σ_1-models are also σ_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

conflict-free pair (x, y)	$x \sqsubseteq \mathcal{O}''(x, y) \sqsubseteq y$
M-conflict-free pair (x, y)	$x \sqsubseteq \mathcal{O}''(x, y) \sqsubseteq y$ and (x, y) is $\leq_i$-maximal
L-conflict-free pair (x, y)	$x \sqsubseteq \mathcal{O}''(x, y) \sqsubseteq y$ and $y \sqcap x^{-1}$ is $\sqsubseteq$-minimal
admissible/reliable pair (x, y)	$(x, y) \leq_i \mathcal{O}(x, y)$
Kripke-Kleene semantics	$lfp(\mathcal{O})$
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	$lfp(\mathcal{SO})$
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 (L-stage, L-supported, L-stable) with additional restrictions on join, meet and involution operations on the lattice.

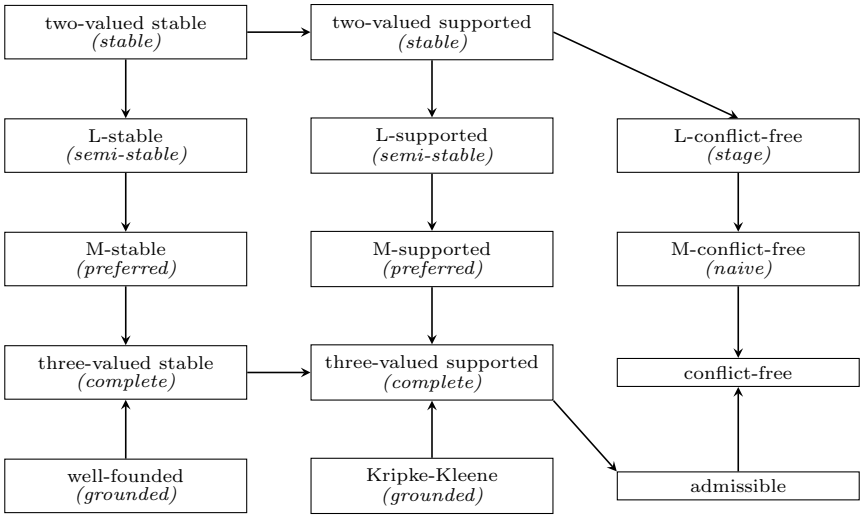


Figure 1: Inclusion relations between operator-based semantics. Nodes depict semantical notions for elements of a bilattice, where the names in parentheses are argumentation versions of these notions. Directed edges indicate subset relationships between the sets of all bilattice elements which satisfy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

and autoepistemic logics via their respective consequence operators [12]. A discussion of these semantics is however out of the scope of this paper.

Finally, Figure 2 on page 40 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 Truszczyński [27], and the translation from default logic into autoepistemic logic of Konolige [26]

Operator	AF	ADF
conflict-free pair	conflict-free set/lab.	conflict-free set/**pair**
M-conflict-free pair	naive extension	**M-conflict-free pair**
L-conflict-free pair	stage extension	**L-conflict-free pair**
reliable 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**. Most 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).

– for details see Denecker et al. [11]. In this paper, we added argumentation frameworks and abstract dialectical frameworks to the picture.

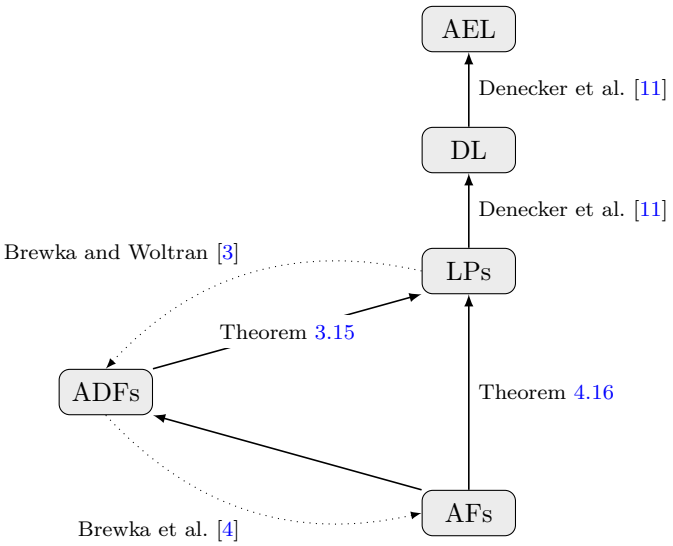


Figure 2: Relative expressiveness of NMR formalisms. Nodes depict nonmonotonic knowledge representation formalisms; argumentation frameworks (AFs), abstract dialectical frameworks (ADFs), logic programs (LPs), default logic (DL) and autoepistemic logic (AEL), respectively. A solid directed edge expresses 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.

Related work. The several new correspondence results for AFs and logic programs we proved extended results of Wu et al. [43], who showed correspondence of complete extensions and three-valued stable models. While the results of Wu et al. [43] use the translation of Gabbay and d'Avila Garcez [19], 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 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 a number of authors, albeit with different goals: Both Wakaki and Nitta [42] and Egly et al. [15] 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. Toni and Sergot [39] survey these and other uses of (forms of) answer set programming to implement abstract argumentation semantics. Nieves et al. [31] define new argumentation semantics that possess certain desired properties. They do this by first providing a general recursive schema for obtaining new logic programming semantics and then defining the argumentation semantics via the AFs' translated logic programs. The translation from AFs into LPs they use is very similar to the one of Dung [14].

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 [23] investigated fixpoint-based definitions of argumentation semantics to study the connection between argumentation and dynamic epistemic logic. Ellmauthaler and Wallner [17] 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 entirely clear how to define the union of two ADFs that share statements. Although for specific representations of acceptance conditions such a union should be straightforward to define, we want to devote some future work into abstracting from specific representations and develop a general method for combining ADFs.

Corollary 3.13 has shown that Brewka and Woltran [3] defined not only the notion of an ADF model, but also the ultimate three-valued approximation of this notion. In recent related work, Brewka et al. [5] introduced several new *ultimate* semantics for abstract dialectical frameworks. In future research we will compare the different semantics defined by their work and in this paper. Furthermore, Denecker et al. [13] 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 and by Brewka et al. [5].

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 KR formalisms. Additionally, in the same way we defined several semantics for normal logic programs (conflict-free, admissible, naive, stage). In order to determine whether these semantics are new, it might be a good starting point to compare them to the semantics discussed by Eiter et al. [16]. Finally, it is of course in order to perform an analysis of the computational complexity of the newly introduced ADF semantics.

Acknowledgements. The author wishes to thank Gerhard Brewka, Stefan Ellmauthaler and Johannes Peter Wallner for useful discussions and providing several (counter-)examples. He is also

grateful to several anonymous referees for providing valuable feedback. This work was partially supported by DFG under project BR-1817/7-1.

[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*, volume 216 of *Frontiers in Artificial Intelligence and Applications*, pages 3–12. IOS Press, September 2010.

[3] Gerhard Brewka and Stefan Woltran. Abstract Dialectical Frameworks. In *Proceedings of the Twelfth International Conference on the Principles of Knowledge Representation and Reasoning (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 *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, pages 780–785, 2011.

[5] Gerhard Brewka, Stefan Ellmauthaler, Hannes Strass, Johannes Peter Wallner, and Stefan Woltran. Abstract Dialectical Frameworks Revisited. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, pages 803–809. AAAI Press, August 2013.

[6] 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.

[7] Martin Caminada. An Algorithm for Stage Semantics. In *Computational Models of Argument: Proceedings of COMMA 2010*, pages 147–158, 2010.

[8] Martin Caminada and Leila Amgoud. On the evaluation of argumentation formalisms. *Artificial Intelligence*, 171(5–6):286–310, 2007.

[9] Claudette Cayrol and Marie-Christine Lagasquie-Schiex. Bipolar abstract argumentation systems. In Guillermo Simari and Iyad Rahwan, editors, *Argumentation in Artificial Intelligence*, pages 65–84. Springer, 2009.

[10] Keith L. Clark. Negation as Failure. In Hervé Gallaire and Jack Minker, editors, *Logic and Data Bases*, pages 293–322. Plenum Press, 1978.

[11] 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.

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

[13] 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.

[14] 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.

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

[16] Thomas Eiter, Michael Fink, and João Moura. Paracoherent Answer Set Programming. In *Proceedings of the Twelfth International Conference on Principles of Knowledge Representation and Reasoning*, May 2010.

[17] Stefan Ellmauthaler and Johannes Peter Wallner. Evaluating Abstract Dialectical Frameworks with ASP. In *Computational Models of Argument: Proceedings of COMMA 2012*, pages 505–506, 2012.

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

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

[20] Michael Gelfond and Vladimir Lifschitz. The Stable Model Semantics for Logic Programming. In *Proceedings of the International Conference on Logic Programming (ICLP)*, pages 1070–1080. The MIT Press, 1988.

[21] 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.

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

[23] Davide Grossi. Fixpoints and Iterated Updates in Abstract Argumentation. In *Proceedings of the Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning (KR)*, pages 65–74. AAAI Press, 2012.

[24] Hadassa Jakobovits and Dirk Vermeir. Robust Semantics for Argumentation Frameworks. *Journal of Logic and Computation*, 9(2):215–261, 1999.

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

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

[27] 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.

[28] Sanjay Modgil. Reasoning about preferences in argumentation frameworks. *Artificial Intelligence*, 173(9–10):901–934, 2009.

[29] Robert Moore. Semantical Considerations of Nonmonotonic Logic. *Artificial Intelligence*, 25(1):75–94, 1985.

[30] Søren H. Nielsen and Simon Parsons. A generalization of Dung's abstract framework for argumentation: Arguing with sets of attacking arguments. In *Argumentation in Multi-Agent Systems*, volume 4766 of *LNCS*, pages 54–73. Springer, 2006.

[31] Juan Carlos Nieves, Mauricio Osorio, and Claudia Zepeda. A Schema for Generating Relevant Logic Programming Semantics and its Applications in Argumentation Theory. *Fundamenta Informaticae*, 106(2–4):295–319, 2011.

[32] Monica Nogueira, Marcello Balduccini, Michael Gelfond, Richard Watson, and Matthew Barry. An A-Prolog decision support system for the Space Shuttle. In Alessandro Provetti and Tran Cao Son, editors, *Answer Set Programming*, 2001.

[33] Henry Prakken. An abstract framework for argumentation with structured arguments. *Argument & Computation*, 1(2):93–124, 2010.

[34] Henry Prakken and Giovanni Sartor. A System for Defeasible Argumentation, with Defeasible Priorities. In *Artificial Intelligence Today*, LNAI, pages 365–379. Springer-Verlag Berlin Heidelberg, 1999.

[35] Raymond Reiter. A Logic for Default Reasoning. *Artificial Intelligence*, 13:81–132, 1980.

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

[37] Hannes Strass. Instantiating knowledge bases in Abstract Dialectical Frameworks. In *Proceedings of the Fourteenth International Workshop on Computational Logic in Multi-Agent Systems (CLIMA XIV)*, volume 8143 of *LNCS*, pages 86–101. Springer, September 2013.

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

[39] Francesca Toni and Marek Sergot. Argumentation and answer set programming. In M. Balduccini and T. Son, editors, *Logic Programming, Knowledge Representation, and Nonmonotonic Reasoning: Essays in Honor of Michael Gelfond*, volume 6565 of *LNAI*, pages 164–180. Springer, 2011.

[40] Bas Van Gijzel and Henry Prakken. Relating Carneades with abstract argumentation. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence – Volume Two*, pages 1113–1119. AAAI Press, 2011.

[41] 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.

[42] Toshiko Wakaki and Katsumi Nitta. Computing Argumentation Semantics in Answer Set Programming. In *Proceedings of the Annual Conference of Japanese Society for Artificial Intelligence (JSAI)*, pages 254–269, 2008.

[43] 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.

[44] Adam Wyner, Trevor Bench-Capon, and Paul Dunne. Instantiating knowledge bases in abstract argumentation frameworks. In *Proceedings of the AAAI Fall Symposium – The Uses of Computational Argumentation*, 2009.