Stable Model Semantics of Abstract Dialectical Frameworks Revisited: A Logic Programming Perspective*

Mario Alviano
University of Calabria, Italy
alviano@mat.unical.it

Wolfgang Faber
University of Huddersfield, UK
wf@wfaber.com

Abstract

This paper relates two extensively studied formalisms: abstract dialectical frameworks and logic programs with generalized atoms or similar constructs. While the syntactic similarity is easy to see, also a strong relation between various stable model semantics proposed for these formalisms is shown by means of a unifying framework in which these semantics are restated in terms of program reducts and an immediate consequence operator, where program reducts have only minimal differences. This approach has advantages for both formalisms, as for example implemented systems for one formalism are usable for the other, and properties such as computational complexity do not have to be rediscovered. As a first, concrete result of this kind, one stable model semantics based on program reducts and subset-minimality that reached a reasonable consensus for logic programs with generalized atoms provides a novel, alternative semantics for abstract dialectical frameworks.

1 Introduction

Abstract argumentation frameworks (AFs; Dung 1995) are used to represent and reason about the fundamental mechanism humans use in argumentation. A well-known example is the Nixon diamond, whose arguments "Nixon is antipacifist since he is a republican" and "Nixon is a pacifist since he is a quaker" attack each other. AFs are represented by graphs whose nodes represent abstract arguments, where abstract refers to the fact that the content of these arguments is not further analyzed. What is analyzed instead is the attack relation, represented by links: an argument is accepted if it is not attacked by any accepted arguments.

Abstract dialectical frameworks (ADFs; Brewka and Woltran 2010) are extensions of AFs that essentially allow for more expressive acceptance conditions. In fact, each statement, or *argument*, of an ADF is associated with a Boolean function whose domain is the power set of the set of parent nodes of the statement. Intuitively, a set of parent nodes is mapped to true if their acceptance jointly supports the statement. Conversely, a set of parent nodes is mapped to false if their acceptance jointly attacks the statement. As an example, consider a scenario in which three lazy programmers a, b, c are asked to accomplish a specific task. Programmer a will work on the task if and only if she can take all the glory (i.e., if neither b nor c will work) or if the total amount of work is minimal (i.e., if both b and c will work as well). Programmer b will work if and only if the amount of work is reasonable, i.e., if a or c will work as well. Programmer c will work if and only if also b will work, regardless of what a will do. Boolean functions can be represented by propositional well-formed formulas, which suggests a syntactic similarity with logic programs with generalized atoms, referred to as logic programs with Boolean functions (LPBFs) in this paper.

Several semantics have been proposed for ADFs, among them stable models [Brewka *et al.*, 2013; Strass, 2013]. Roughly, a model is considered stable if it can be reconstructed via *logical entailment* in the *reduced* ADF obtained by constraining the original acceptance conditions. Different notions of logical entailment and reduct result in different stable model semantics. Stable models are also defined for LPBFs [Pelov *et al.*, 2007; Son and Pontelli, 2007; Shen and Wang, 2012; Gelfond and Zhang, 2014; Faber *et al.*, 2011; Ferraris, 2005]. A natural question is thus "What relations do exist between these notions of stable models?"

This paper provides an answer to this question by introducing mappings from the two formalisms to what will be referred to as *definitorial* LPBFs, i.e., LPBFs in which each propositional atom is defined by exactly one rule whose body consists of exactly one Boolean function. By restating the original definitions of stable models, the paper will shed light on the similarities and differences between these semantics. In particular, stable models by Brewka *et al.* are equivalent to those by Pelov *et al.*, Son and Pontelli and by Shen and Wang, while stable models by Strass are equivalent to those by Gelfond and Zhang. The new, restated definitions will also demonstrate a subset-containment relation between these semantics: every stable model of Gelfond and Zhang is a stable model of Son and Pontelli, and the inclusion is strict as the

*Mario Alviano was partly supported by MIUR within project "SI-LAB BA2KNOW – Business Analitycs to Know", by Regione Calabria, POR Calabria FESR 2007-2013, within project "ITravel PLUS" and project "KnowRex", by the National Group for Scientific Computation (GNCS-INDAM), and by Finanziamento Giovani Ricercatori UNICAL.

other direction does not hold in general.

A further advantage of the unifying framework presented in this paper is that implemented systems for LPBFs are usable for ADFs, and vice versa. For example, the answer set programming (ASP) solvers CLASP [Gebser *et al.*, 2009] and DLV [Alviano *et al.*, 2011] compute stable models as defined by Ferraris and Faber *et al.*, respectively, which are known to be equivalent to the definition by Pelov *et al.*, Son and Pontelli and Shen and Wang for a large class of Boolean functions known as convex [Alviano and Faber, 2013], and therefore also to the notion by Brewka *et al.* thanks to the result in this paper. Hence, for ADFs whose accepting conditions can be expressed by standard aggregates, a one-to-one mapping to the input language of CLASP is possible. On the other hand, the rewritings implemented in DIAMOND [Brewka *et al.*, 2013; Ellmauthaler and Strass, 2014] to map ADFs to ASP can be used in ASP as well to encode convex Boolean functions other than standard aggregates. The simplicity of the new characterizations also suggests a strategy for computing stable models by Strass and by Gelfond and Zhang within an ASP solver. Moreover, the unifying framework brings several complexity results from one formalism to the other, possibly making the future complexity analysis even more attractive as new results will immediately apply to both LPBFs and ADFs. Finally, stable models by Faber *et al.* and Ferraris, popular in LPBFs but having no counterpart in ADFs, are also restated in terms of the unifying framework. The lazy programmers example will be used to illustrate what differences there are to the existing semantics, and why stable models by Faber *et al.* and by Ferraris can be considered a valid point of view also for ADFs. Proofs and a tool for computing Strass stable models by modern ASP solvers such as CLASP [Gebser *et al.*, 2009] or WASP [Alviano *et al.*, 2013; 2014] are available at http://alviano.net/software/adf/.

2 Background

This section introduces basic notions concerning ADFs and LPBFs.

2.1 Boolean Functions and Interpretations

Let **T**,**U**,**F** denote the truth values true, undefined and false, respectively. Let $\mathcal{V}$ be a set of propositional variables, also referred to as statements or atoms. A Boolean function f is a function $f : 2^D \to \{\mathbf{T}, \mathbf{F}\}$, where $D \subseteq \mathcal{V}$ defines the domain of f. For such a function f, the set D is denoted by $dom(f)$, and $f(V)$ will be used as a shorthand for $f(dom(f) \cap V)$, for all $V \subseteq \mathcal{V}$.

Example 1. Simple Boolean functions are represented by conjunctions, disjunctions, and in general well-formed propositional formulas. In fact, the conjunction $a \wedge (b \vee c)$ defines a function f such that $dom(f) = \{a, b, c\}$, $f(\{a, b\}) = f(\{a, c\}) = f(\{a, b, c\}) = \mathbf{T}$, and $f(\emptyset) = f(\{a\}) = f(\{b\}) = f(\{c\}) = f(\{b, c\}) = \mathbf{F}$. Also aggregates define Boolean functions: $COUNT(\{a, b\}) = 1$ defines a function f such that $dom(f) = \{a, b\}$, $f(\{a\}) = f(\{b\}) = \mathbf{T}$, and $f(\emptyset) = f(\{a, b\}) = \mathbf{F}$. ■

An *interpretation* is a pair (L, U) such that $L \subseteq \mathcal{V}$ and $U \subseteq \mathcal{V}$, where L, U are called lower and upper bound, respectively. This is based on a common representation of four-valued interpretations that stores true atoms (L) and possibly true atoms (U). Intuitively, atoms in L are true, those in $U \setminus L$ are undefined, those in $\mathcal{V} \setminus U$ are false, and those in $L \setminus U$ are inconsistent. An interpretation (L', U') *extends* an interpretation (L, U), denoted $(L, U) \leq_k (L', U')$, if $L' \supseteq L$ and $U' \subseteq U$. Relation $\leq_k$ corresponds to the knowledge order, i.e., undefined atoms do not increase in an extension. An interpretation (L, U) is *total* or two-valued if $L = U$; in this case, the interpretation will be usually denoted L. Two sets S, S' of total interpretations are equivalent with respect to a context $V \subseteq \mathcal{V}$, denoted $S \equiv_V S'$, if $\{I \cap V \mid I \in S\} = \{I \cap V \mid I \in S'\}$.

A Boolean function f is true with respect to an interpretation (L, U) if $f(I) = \mathbf{T}$ for all $I \subseteq \mathcal{V}$ such that $(L, U) \leq_k I$. Similarly, f is false if $f(I) = \mathbf{F}$ for all $I \subseteq \mathcal{V}$ such that $(L, U) \leq_k I$. If none of the previous cases apply then f is considered undefined. These concepts are formalized next.

Definition 1. The evaluation of a Boolean function f with respect to an interpretation (L, U) is defined as follows:

$$[f]_{(L,U)} := \begin{cases} \mathbf{T} & \text{if } f(I) = \mathbf{T} \text{ for all } L \subseteq I \subseteq U \\ \mathbf{U} & \text{if } f(I) = \mathbf{T} \text{ and } f(J) = \mathbf{F} \text{ for} \\ & \quad \text{some } L \subseteq I \subseteq U, L \subseteq J \subseteq U \\ \mathbf{F} & \text{otherwise.} \end{cases} \quad (1)$$

(Note that $L \not\subseteq U$ implies $[f]_{(L,U)} = \mathbf{T}$ for all functions.)

2.2 Abstract Dialectical Frameworks

An *abstract dialectical framework* F is a triple $\langle \mathcal{S}, E, C \rangle$, where: $\mathcal{S}$ is a nonempty, finite set of statements; $E \subseteq \mathcal{S} \times \mathcal{S}$ is a set of links; $C = \{C_s\}_{s \in \mathcal{S}}$ is a set of Boolean functions, where $dom(C_s) = par(s)$, with $par(s)$ denoting the parents of s in the graph $\langle \mathcal{S}, E \rangle$. Each C_s is referred to as the *acceptance condition* of s.

A total interpretation I is a model of F, denoted $I \models F$, if $s \in I$ whenever $[C_s]_I = \mathbf{T}$. A stable model semantics for ADF was proposed by Brewka *et al.* (2013). It is based on the following operator:

$$\Gamma_F(L, U) := (\{s \in \mathcal{S} \mid [C_s]_{(L,U)} = \mathbf{T}\}, \{s \in \mathcal{S} \mid [C_s]_{(L,U)} \neq \mathbf{F}\}) \quad (2)$$

where F is an ADF, and (L, U) is an interpretation. The least fixpoint of Γ_F is denoted by $\Gamma_F \Uparrow (\emptyset, \mathcal{S})$.

Definition 2 (Brewka *et al.*). A total interpretation I is a B-stable model of an ADF $F = \langle \mathcal{S}, E, C \rangle$ if $I \models F$ and $I = \Gamma_{F'} \Uparrow (\emptyset, \mathcal{S})$, where $F' = \langle I, E \cap (I \times I), \{C'_s\}_{s \in I} \rangle$, and C'_s is a Boolean function with domain $par(s) \cap I$ and such that $C'_s(A) = \mathbf{T}$ if and only if both $C_s(A) = \mathbf{T}$ and $(par(s) \setminus A) \cap I = \emptyset$, for all $A \subseteq par(s)$.

A different notion of stable model was proposed by Strass (2013). In particular, the characterization in (the proof of) Proposition 3.9 in [Strass, 2013] is based on the least fixpoint $G^I_F \Uparrow \emptyset$ of the following operator:

$$G^I_F(X) := \{s \in \mathcal{S} \mid \exists A \subseteq X,\ C_s(A) = \mathbf{T},\ (par(s) \setminus A) \cap I = \emptyset,\ (par(s) \setminus A) \cap X = \emptyset\} \quad (3)$$

where $F = \langle \mathcal{S}, E, C \rangle$ is an ADF, I is a total interpretation, and $X \subseteq \mathcal{S}$.

Definition 3 (Strass). A total interpretation I is an S-stable model of an ADF $F = \langle \mathcal{S}, E, C \rangle$ if $I = G_F^I \Uparrow (\emptyset, \mathcal{S})$.

The two semantics are not equivalent.

Example 2. Consider an ADF $F_1 = \langle \mathcal{S}, E, C \rangle$ such that:

- $\mathcal{S} = \{a, b\}$, and $E = \{(a, a), (b, a), (b, b)\}$;
- $C_a(\emptyset) = C_a(\{a\}) = C_a(\{a, b\}) = \mathbf{T}$, $C_a(\{b\}) = \mathbf{F}$;
- $C_b(\{b\}) = \mathbf{T}$, and $C_b(\emptyset) = \mathbf{F}$.

F_1 has one stable model, $\{a\}$, according to Brewka *et al.*. In fact, $\{a\} \models F_1$, and ADF F_1' is $\langle \{a\}, \{(a, a)\}, \{C_a'\} \rangle$, where $dom(C_a') = \{a\}$ and $C_a'(\emptyset) = C_a'(\{a\}) = \mathbf{T}$. Hence, the least fixpoint of $\Gamma_{F_1'}$ is $\{a\}$. On the other hand, F_1 has no stable model according to Strass. In particular, $\{a\}$ is not stable because the least fixpoint of $G_{F_1}^{\{a\}}$ is $\emptyset$. ■

Example 3. Consider an ADF $F_2 = (\mathcal{S}, E, C)$ such that:

- $\mathcal{S} = \{a, b, c\}$, and $E = \{(a, b), (a, c), (b, a), (c, c)\}$;
- $C_a(\emptyset) = \mathbf{T}$, $C_a(\{b\}) = \mathbf{F}$; $C_b(\emptyset) = \mathbf{T}$, $C_b(\{a\}) = \mathbf{F}$;
- $C_c(\{b\}) = C_c(\{b, c\}) = \mathbf{T}$; $C_c(\emptyset) = C_c(\{c\}) = \mathbf{F}$.

Interpretation $\{a\}$ is a stable model according to both Brewka *et al.* and Strass, while $\{b, c\}$ is a stable model only according to Brewka *et al.* ■

2.3 Logic Programs with Boolean Functions

A *logic program with Boolean functions* Π is a set of rules of the form $p \leftarrow f$, where $p \in \mathcal{V}$ and f is a Boolean function. The set of atoms occurring in Π is denoted by $At(\Pi)$. A total interpretation I is a *model* of Π, denoted $I \models \Pi$, if $p \in I$ whenever $[f]_I = \mathbf{T}$, for all $p \leftarrow f \in \Pi$.

Stable models by Pelov *et al.* (2007), Son and Pontelli (2007) and by Shen and Wang (2012) are defined by means of the least fixpoint $K_\Pi^I \Uparrow \emptyset$ of the following operator:

$$K_\Pi^I(X) := \{p \in \mathcal{V} \mid \exists\, p \leftarrow f \in \Pi, \; [f]_I = \mathbf{T}, \; [f]_{(X,I)} = \mathbf{T}\} \tag{4}$$

where Π is an LPBF, I is a total interpretation, and $X \subseteq \mathcal{V}$.

Definition 4 (Pelov *et al.*, Son and Pontelli, Shen and Wang). A total interpretation I is a stable model of an LPBP Π if $I = K_\Pi^I \Uparrow \emptyset$.

A different notion of stable model was proposed by Gelfond and Zhang (2014).

Definition 5 (Gelfond and Zhang). A total interpretation I is a stable model of an LPBP Π if $I \models \Pi$ and there is no $J \subset I$ such that J is a model of the following program:

$$\{p \leftarrow \bigwedge_{q \in dom(f) \cap I} q \mid [f]_I = \mathbf{T}\}. \tag{5}$$

The two semantics are not equivalent.

Example 4. Let $\Pi_1 = \{a \leftarrow g,\ b \leftarrow h\}$, where:

- $dom(g) = \{a, b\}$, $g(\emptyset) = g(\{a\}) = g(\{a, b\}) = \mathbf{T}$, $g(\{b\}) = \mathbf{F}$;
- $dom(h) = \{b\}$, $h(\{b\}) = \mathbf{T}$, and $h(\emptyset) = \mathbf{F}$.

Π_1 has one stable model according to Son and Pontelli, namely $\{a\}$. In fact, $\{a\}$ is the least fixpoint of $K_\Pi^{\{a\}}$. On the other hand, Π_1 has no stable model according to Gelfond and Zhang. In particular, $\{a\}$ is not a stable model because program (5) is $\Pi_1' = \{a \leftarrow a\}$, and it is satisfied by $\emptyset$. ■

Example 5. Let $\Pi_2 = \{a \leftarrow g,\ b \leftarrow h,\ c \leftarrow o\}$, where:

- $dom(g) = \{b\}$, $g(\emptyset) = \mathbf{T}$, $g(\{b\}) = \mathbf{F}$;
- $dom(h) = \{a\}$, $h(\emptyset) = \mathbf{T}$, $h(\{a\}) = \mathbf{F}$;
- $dom(o) = \{b, c\}$, $o(\{b\}) = o(\{b, c\}) = \mathbf{T}$, and $o(\emptyset) = o(\{c\}) = \mathbf{F}$.

Interpretation $\{a\}$ is a stable model according to Son and Pontelli and Gelfond and Zhang, while $\{b, c\}$ is a stable model only according to Son and Pontelli. ■

3 Definitorial LPBFs

A unifying framework is presented in this section to better understand similarities and differences between the several stable model semantics for ADFs and LPBFs.

A *definitorial logic program with Boolean functions* is an LPBF Π in which there is exactly one rule of the form $p \leftarrow f$ for each $p \in \mathcal{V}$. For a DLPBF Π containing $p \leftarrow f$, function f is denoted Π_p. The *immediate consequence* operator is defined as follows:

$$T_\Pi^U(L) := \{p \in \mathcal{V} \mid [\Pi_p]_{(L,U)} = \mathbf{T}\} \tag{6}$$

where Π is a DLPBF, and (L, U) is an interpretation. The least fixpoint of T_Π^U is denoted by $T_\Pi^U \Uparrow \emptyset$.

The new characterizations require to compute the fixpoint of the immediate consequence operator for a program reduct obtained by modifying Boolean functions of the input program. In more detail, for a total interpretation I, and a Boolean function f, the following functions are defined:

$$keep\_all(f, I)(A) := f(I) \wedge f(A) \tag{7}$$

$$keep\_subsets(f, I)(A) := f(I) \wedge f(A) \wedge (dom(f) \cap A \subseteq dom(f) \cap I) \tag{8}$$

$$keep\_equal(f, I)(A) := f(I) \wedge f(A) \wedge (dom(f) \cap A = dom(f) \cap I) \tag{9}$$

$$keep\_supsets(f, I)(A) := f(I) \wedge f(A) \wedge (dom(f) \cap A \supseteq dom(f) \cap I) \tag{10}$$

for all $A \subseteq dom(f) = dom(k(f, I))$, where $k \in \{keep\_all, keep\_subsets, keep\_equal, keep\_supsets\}$.

Definition 6 (B-stable model). A total interpretation I is a B-stable model of a DLPBF Π if $I = T_{B(\Pi,I)}^I \Uparrow \emptyset$, where:

$$B(\Pi, I) = \{p \leftarrow keep\_subsets(\Pi_p, I) \mid p \in \mathcal{V}\}. \tag{11}$$

Definition 7 (S-stable model). A total interpretation I is an S-stable model of a DLPBF Π if $I = T_{S(\Pi,I)}^I \Uparrow \emptyset$, where:

$$S(\Pi, I) = \{p \leftarrow keep\_supsets(\Pi_p, I) \mid p \in \mathcal{V}\}. \tag{12}$$

Definition 8 (P-stable model). A total interpretation I is a P-stable model of a DLPBF Π if $I = T_{P(\Pi,I)}^I \Uparrow \emptyset$, where:

$$P(\Pi, I) = \{p \leftarrow keep\_all(\Pi_p, I) \mid p \in \mathcal{V}\}. \tag{13}$$

Definition 9 (G-stable model). A total interpretation I is a G-stable model of a DLPBF Π if $I = T^I_{G(\Pi,I)} \Uparrow \emptyset$, where:

$$G(\Pi, I) = \{p \leftarrow \mathit{keep\_equal}(\Pi_p, I) \mid p \in \mathcal{V}\}. \quad (14)$$

Example 6. Program Π_1 from Example 4 is definitorial. Program $B(\Pi_1, \{a\})$ contains the following functions:

- $\mathit{keep\_subsets}(g, \{a\})$, assigning $\mathbf{T}$ only to $\emptyset$ and $\{a\}$;
- $\mathit{keep\_subsets}(h, \{a\})$, always assigning $\mathbf{F}$.

It can be checked that $\{a\}$ is a B-stable model. Program $S(\Pi_1, \{a\})$ contains the following functions:

- $\mathit{keep\_supsets}(g, \{a\})$, assigning $\mathbf{T}$ only to $\{a\}$ and $\{a, b\}$;
- $\mathit{keep\_supsets}(h, \{a\})$, always assigning $\mathbf{F}$.

It can be checked that $\{a\}$ is not an S-stable model. Program $P(\Pi_1, \{a\})$ contains the following functions:

- $\mathit{keep\_all}(g, \{a\})$, assigning $\mathbf{T}$ only to $\emptyset$, $\{a\}$ and $\{a, b\}$;
- $\mathit{keep\_all}(h, \{a\})$, always assigning $\mathbf{F}$.

It can be checked that $\{a\}$ is a P-stable model. Program $G(\Pi_1, \{a\})$ contains the following functions:

- $\mathit{keep\_equal}(g, \{a\})$, assigning $\mathbf{T}$ only to $\{a\}$;
- $\mathit{keep\_equal}(h, \{a\})$, always assigning $\mathbf{F}$.

It can be checked that $\{a\}$ is not a G-stable model. Moreover, it can be checked that there is no other stable model. ■

Example 7. Program Π_2 from Example 5 is definitorial. Interpretation $\{a\}$ is a B-, S-, P- and G-stable model. Interpretation $\{b, c\}$ is a B- and P-stable model. The program has no other stable model according to the previous definitions. ■

4 Equivalences

Examples 6–7 suggest that Definitions 4 and 8 are equivalent to Definitions 5 and 9, respectively.

Theorem 1. *Let Π be a DLPBF, S be the set of stable models of Π according to Son and Pontelli (2007), and S' be the set of P-stable models of Π, then $S \equiv_{At(\Pi)} S'$.*

Theorem 2. *Let Π be a DLPBF, S be the set of stable models of Π according to Gelfond and Zhang (2014), and S' be the set of G-stable models of Π, then $S \equiv_{At(\Pi)} S'$.*

DLPBFs can model ADFs under B- and S-stable model semantics. In fact, an ADF $F = \langle \mathcal{S}, E, C \rangle$ can be transformed into a DLPBF $\mathit{def}(F)$ comprising the following rules:

- $s \leftarrow C_s$, for all $s \in \mathcal{S}$;
- $p \leftarrow \bot$, where $\bot$ is such that $\mathit{dom}(\bot) = \emptyset$ and $\bot(\emptyset) = \mathbf{F}$, for all $p \in \mathcal{V} \setminus \mathcal{S}$.

Example 8. Consider the ADFs F_1, F_2 from Examples 2–3, and the DLPBFs Π_1, Π_2 from Examples 4–5. Note that $\mathit{def}(F_1) = \Pi_1$ and $\mathit{def}(F_2) = \Pi_2$ (if rules of the form $p \leftarrow \bot$ are added for all p not occurring in F_1, F_2). ■

Equivalence of Definitions 2 and 6, and of Definitions 3 and 7, can now be established.

Theorem 3. *Let $F = \langle \mathcal{S}, E, C \rangle$ be an ADF, S be the set of stable models of F according to Brewka* et al. *(2013), and S' be the set of B-stable models of $\mathit{def}(F)$, then $S \equiv_{\mathcal{S}} S'$.*

Theorem 4. *Let $F = \langle \mathcal{S}, E, C \rangle$ be an ADF, S be the set of stable models of F according to Strass (2013), and S' be the set of S-stable models of $\mathit{def}(F)$, then $S \equiv_{\mathcal{S}} S'$.*

The unifying framework also provides a better overview of the four stable model semantics, which may also help to better understand their similarities and differences. The examples in the previous sections already gave hints to relationships between the various stable model semantics. In the following, we motivate and state the general results.

The reducts used by B- and P-stable models replace each Boolean function f with $\mathit{keep\_subsets}(f, I)$ and $\mathit{keep\_all}(f, I)$, respectively. The immediate consequence operator is then applied on the reducts, with the upper bound fixed to I. Hence, the image of $\mathit{keep\_all}(f, I)$ on any $V \supset \mathit{dom}(f)$ is irrelevant for the fixpoint computation.

Theorem 5. *A total interpretation I is a B-stable model of a DLPBF Π if and only if I is a P-stable model of Π.*

Similarly, the reducts used by S- and G-stable models replace each Boolean function f with $\mathit{keep\_supsets}(f, I)$ and $\mathit{keep\_equal}(f, I)$, respectively. Again, the image of $\mathit{keep\_supsets}(f, I)$ on any $V \supset \mathit{dom}(f)$ is irrelevant for the fixpoint computation.

Theorem 6. *A total interpretation I is an S-stable model of a DLPBF Π if and only if I is a G-stable model of Π.*

Concerning P- and G-stable models, their reducts use $\mathit{keep\_all}(f, I)$ and $\mathit{keep\_equal}(f, I)$, respectively. Any atom inferred by the immediate consequence operator on the reduct using $\mathit{keep\_equal}(f, I)$ is also inferred when $\mathit{keep\_all}(f, I)$ is used, while the converse does not hold in general.

Theorem 7. *If a total interpretation I is a G-stable model of a DLPBF Π then I is a P-stable model of Π.*

The following is a corollary of the previous theorems, and can also be obtained by combining results in approximation fixpoint theory [Denecker *et al.*, 2004; Pelov *et al.*, 2007; Strass and Wallner, 2014].

Corollary 1. *If a total interpretation I is an S-stable model of a DLPBF Π then I is a B-stable model of Π.*

5 Compilations

S- or G-stable models of a DLPBF Π can be obtained by means of B- or P-stable models of a rewritten DLPBF $\mathit{s2b}(\Pi)$. In more detail, $\mathit{At}(\mathit{s2b}(\Pi)) = \{p, p^F, p^= \mid p \in \mathit{At}(\Pi)\} \cup \{\bot\}$, and $\mathit{s2b}(\Pi)$ consists of the following rules:

- $p \leftarrow f_p$, where $\mathit{dom}(f_p) = \{p^F\}$, $f_p(\emptyset) = \mathbf{T}$ and $f_p(\{p^F\}) = \mathbf{F}$;
- $p^F \leftarrow g_p$, where $\mathit{dom}(g_p) = \{p\}$, $g_p(\emptyset) = \mathbf{T}$ and $g_p(\{p\}) = \mathbf{F}$;
- $p^= \leftarrow h_p$, where $\mathit{dom}(h_p) = \{p^F\} \cup \{q^= \mid q \in \mathit{dom}(\Pi_p)\}$, and $h_p(I) = \mathbf{T}$ if and only if either $p^F \in I$, or both $\Pi_p(I) = \mathbf{T}$ and $\{q^= \mid q \in \mathit{dom}(\Pi_p)\} \subseteq I$;

- $\bot \leftarrow o$, where $dom(o) = At(s2b(\Pi))$, and $s2b(I) = \mathbf{T}$ if and only if $\bot \notin I$ and there is $p \leftarrow f \in \Pi$ such that either $p^{=} \notin I$, or both $f(I) = \mathbf{T}$ and $p^F \in I$.

Intuitively, atoms p, p^F are used to guess an interpretation $I \cap At(\Pi)$ for Π. Since atom $\bot$ must be false in all B-stable models of $s2b(\Pi)$, $I \cap At(\Pi)$ must actually represent a model of Π if I is a B-stable model of $s2b(\Pi)$. Moreover, the falsity of $\bot$ also implies that all atoms of the form $p^{=}$ must be true. However, each $p^{=}$ can be derived by the immediate consequence operator only if either $p \notin I$, or if all atoms $q^{=}$ such that $q \in dom(\Pi_p)$ are already derived in a previous application of the operator.

Theorem 8. *Let Π be a DLPBF, S be the set of S- or G-stable models of Π, and S' be the set of B- or P-stable models of $s2b(\Pi)$, then $S \equiv_{At(\Pi)} S'$.*

An ASP encoding based on the rewriting $s2b(\cdot)$ and allowing for computing S- or G-stable models of an ADF encoded according to the DIAMOND format is available online.

Further compilations are possible. For example, for stable models by Son and Pontelli (2007), an LPBF Π can be transformed into a DLPBF $def(\Pi)$ comprising the following rules:

- $aux_f \leftarrow f$, where $aux_f \in \mathcal{V} \setminus At(\Pi)$ is an auxiliary atom associated with function f, for all $p \leftarrow f \in \Pi$;
- $p \leftarrow \bigvee_{p \leftarrow f \in \Pi} aux_f$, for all $p \in At(\Pi)$.

Example 9. Let g, h, o be Boolean functions such that: $dom(g) = \{b\}$, $g(\emptyset) = \mathbf{T}$, $g(\{b\}) = \mathbf{F}$; $dom(h) = \{b\}$, $h(\{b\}) = \mathbf{T}$, $h(\emptyset) = \mathbf{F}$; $dom(o) = \{a\}$, $o(\{a\}) = \mathbf{T}$, $o(\emptyset) = \mathbf{F}$. Let $\Pi_3 = \{a \leftarrow g,\ a \leftarrow h\}$. Essentially, program Π_3 is $\{a \leftarrow not\ b,\ a \leftarrow b\}$, where not is the negation as failure symbol under stable model semantics [Gelfond and Lifschitz, 1988; 1991]. Program $def(\Pi_3)$ comprises the following rules: $\{a \leftarrow f_a,\ aux_g \leftarrow g,\ aux_h \leftarrow h\}$, where f_a is the disjunction $aux_g \vee aux_h$, i.e., $dom(f_a) = \{aux_g, aux_h\}$ and f_a assigns $\mathbf{F}$ only to $\emptyset$. It can be noted that $\{a\}$ is the unique stable model of Π_3 according to Definition 4. Similarly, $\{a\} \cup \{aux_g\}$ is the unique stable model of $def(\Pi_3)$ according to Definition 8.

Consider now $\Pi_4 = \Pi_3 \cup \{b \leftarrow o\}$, i.e., $\{a \leftarrow not\ b,\ a \leftarrow b,\ b \leftarrow a\}$. Program $def(\Pi_4)$ comprises the following rules: $\{a \leftarrow f_a,\ aux_g \leftarrow g,\ aux_h \leftarrow h,\ b \leftarrow f_b,\ aux_o \leftarrow o\}$, where f_b is the disjunction aux_o, i.e., $dom(f_b) = \{aux_o\}$ and f_o assigns $\mathbf{F}$ only to $\emptyset$. It can be noted that Π_4 and $def(\Pi_4)$ have no stable model according to Definitions 6–9.

Note that a rewriting introducing rules of the form

$$p \leftarrow \bigvee_{p \leftarrow f \in \Pi} f \tag{15}$$

would possibly change the semantics of the program. In fact, the rewriting of program Π_4 would be $\{a \leftarrow g \vee h,\ b \leftarrow o\}$, where $dom(g \vee h) = \{b\}$ and $g \vee h$ assigns $\mathbf{T}$ to both $\emptyset$ and $\{b\}$. Interpretation $\{a, b\}$ is now a B- and P-stable model. ■

Theorem 1 can thus be extended to LPBFs in general.

Theorem 9. *Let Π be an LPBF, S be the set of stable models of Π according to Son and Pontelli (2007), and S' be the set of P-stable models of $def(\Pi)$, then $S \equiv_{At(\Pi)} S'$.*

Concerning stable models by Gelfond and Zhang (2014), rewriting $s2b(\cdot)$ can be extended to LPBFs by replacing each function h_p occurring in rules of the form $p^{=} \leftarrow h_p$ as follows: $dom(h_p) = \{p^F\} \cup \{q^{=} \mid \exists p \leftarrow f \in \Pi,\ q \in dom(f)\}$, and $h_p(I) = \mathbf{T}$ if and only if either $p^F \in I$, or there is $p \leftarrow f \in \Pi$ such that both $f(I) = \mathbf{T}$ and $\{q^{=} \mid q \in dom(f)\} \subseteq I$.

Theorem 10. *Let Π be an LPBF, S be the set of stable models of Π according to Gelfond and Zhang (2014), and S' be the set of B- or P-stable models of $s2b(\Pi)$, then $S \equiv_{At(\Pi)} S'$.*

It is also true that a DLPBF Π can be transformed into an ADF $adf(\Pi) = \langle At(\Pi), E, C \rangle$, where $E = \{(p, q) \mid q \in At(\Pi),\ p \in dom(\Pi_q)\}$, and $C = \{\Pi_p \mid p \in At(\Pi)\}$.

Example 10. Consider again the ADFs F_1, F_2 from Examples 2–3, and the DLPBFs Π_1, Π_2 from Examples 4–5. Note that $adf(\Pi_1) = F_1$ and $adf(\Pi_2) = F_2$. ■

As done before, equivalence of Definitions 2 and 6, and of Definitions 3 and 7, can be established.

Theorem 11. *Let Π be a DLPBF, S be the set of B-stable models of Π, and S' be the set of stable models of $adf(\Pi)$ according to Brewka* et al.*, then $S \equiv_{\mathcal{S}} S'$.*

Theorem 12. *Let Π be a DLPBF, S be the set of S-stable models of Π, and S' be the set of stable models of $adf(\Pi)$ according to Strass, then $S \equiv_{\mathcal{S}} S'$.*

By combining the previous transformations, an LPBF Π can be transformed into an ADF $adf(def(\Pi))$.

Corollary 2. *Let Π be an LPBF, S be the set of stable models of Π according to Brewka* et al.*, and S' be the set of stable models of $adf(def(\Pi))$ according to Brewka* et al.*, then $S \equiv_{At(\Pi)} S'$.*

Corollary 3. *Let Π be an LPBF, S be the set of stable models of Π according to Strass, and S' be the set of stable models of $adf(s2b(\Pi))$ according to Brewka* et al.*, then $S \equiv_{At(\Pi)} S'$.*

Example 11. Consider program Π_4 from Example 9. The ADF $adf(def(\Pi_4)) = \langle \mathcal{S}, E, C \rangle$ is such that: $\mathcal{S} = \{a, aux_g, aux_h, b, aux_o\}$; $E = \{(aux_g, a),\ (aux_h, a),\ (b, aux_g),\ (b, aux_h),\ (aux_o, b),\ (a, aux_o)\}$; $C_a = \Pi_a$, $C_{aux_g} = g$, $C_{aux_h} = h$, $C_b = \Pi_b$, $C_{aux_o} = o$. Note that $adf(def(\Pi_4))$ has no stable model according to all definitions. As observed in Example 9, transforming Π_4 into an ADF $\langle \{a, b\}, \{(b, a), (a, b)\}, C \rangle$ such that $C_a(\emptyset) = C_a(\{b\}) = \mathbf{T}$, $C_b(\{a\}) = \mathbf{T}$, and $C_b(\emptyset) = \mathbf{F}$, would make $\{a\}$ a stable model according to Brewka *et al.* ■

6 F-Stable Models

There are cases in which the previously considered semantics may be considered too restrictive.

Example 12. The lazy programmers example in the introduction can be represented by the DLPBF $\Pi_5 = \{a \leftarrow g,\ b \leftarrow h,\ c \leftarrow o\}$, where: $dom(g) = \{b, c\}$, $dom(h) = \{a, c\}$, $dom(o) = \{b\}$; $g(\emptyset) = g(\{b, c\}) = \mathbf{T}$, $g(\{b\}) = g(\{c\}) = \mathbf{F}$; $h(\{a\}) = h(\{c\}) = h(\{a, c\}) = \mathbf{T}$, $h(\emptyset) = \mathbf{F}$; $o(\{b\}) = \mathbf{T}$, $o(\emptyset) = \mathbf{F}$. There are no stable models according to all previous definitions. As an example, consider the reducts for $I = \{a, b, c\}$: $B(\Pi_5, I) = P(\Pi_5, I) = \Pi_5$;

$S(\Pi_5, I) = G(\Pi_5, I) = \{a \leftarrow f_a,\ b \leftarrow f_b,\ c \leftarrow o\}$, where f_a and f_b are equivalent to $b \wedge c$ and $a \wedge c$, respectively. In all cases, the least fixpoints are equal to the empty set. ■

Other semantics for LPBFs were proposed and analyzed in the previous decade. Two of them, known as FLP [Faber *et al.*, 2011] and Ferraris stable models [Ferraris, 2005], are implemented by current ASP solvers [Faber *et al.*, 2008; Gebser *et al.*, 2009]. These two semantics are equivalent for negation-free programs, and are thus equivalent for DLPBFs. The following is a restatement of the original definitions of stable models by Faber *et al.* and by Ferraris.

Definition 10 (F-stable model). A total interpretation I is an F-stable model of a DLPBF Π if $I \models \Pi$ and there is no $J \subset I$ such that $J \models F(\Pi, I)$, where:

$$F(\Pi, I) = \{p \leftarrow keep\_all(\Pi_p, I) \mid p \in \mathcal{V}\}. \qquad (16)$$

The following link follows from Theorem 5 in [Shen and Wang, 2012], Definition 3 in [Son *et al.*, 2007] and Theorem 2 in [Son and Pontelli, 2007].

Corollary 4. *Let Π be a DLPBF, and I be a total interpretation. If I is a G-, S-, P- or B-stable model then I is an F-stable model.*

The other direction does not hold in general.

Example 13. Recall Π_5, the lazy programmers example. As shown in Example 12, there are no B-, S-, P- or G-stable models. However, it does have an F-stable model, $\{a, b, c\}$. Indeed, the reduct $F(\Pi_5, \{a, b, c\})$ equals Π_5, and it can be checked that $\{a, b\}$ and $\{b\}$ do not satisfy the rule defining c, while $\{a, c\}$, $\{a\}$, $\{c\}$ do not satisfy the rule defining b, and $\{b, c\}$ and $\emptyset$ do not satisfy the rule defining a. ■

It is instructive to examine Π_5 a bit more closely. It illustrates the different underlying principles of F-stable and the other notions of stable models. F-stable models also allow for "jumping over gaps," while the immediate consequence operator is not allowed to make an inference if any interpretation between the fixed upper bound and the current lower bound evaluates to false. This is the case in Π_5, as there are total interpretations between $\emptyset$ and $\{a, b, c\}$ for which g evaluates to false (these are $\{b\}$, $\{c\}$, $\{a, b\}$, and $\{a, c\}$). However, it is not clear why the presence of these interpretations should inhibit $\{a, b, c\}$ to be a stable model, as they are never reached during the computation of the fixpoint. Brewka *et al.* (2013) argue for a similar example that the justification is cyclic. This depends on the point of view: in terms of Π_5, the cyclic justification claimed by Brewka *et al.* would be as follows: a is true because b and c are true, while b is true because a is true, and c is true because b is true. However, one can argue as well that a does have also another, non-circular justification that grounds the cycle, namely $\emptyset$. Indeed, in terms of the lazy programmer example, one could imagine a dynamic process as follows: if initially nobody works, programmer a will decide to work, which makes programmer b decide to work, causing also programmer c to work. While the original reason for programmer a has been invalidated by this, she still finds herself in a situation that supports the initial decision.

7 Related Work

For ADFs, S-stable models [Strass, 2013] and B-stable models [Strass and Wallner, 2014] were characterized using approximation fixpoint theory [Denecker *et al.*, 2004]. Approximation fixpoint theory was also used to characterize some stable model semantics for LPBFs [Pelov *et al.*, 2007]. The containment relation between stable models by Strass and by Brewka *et al.*, i.e., Corollary 1 in Section 4, can be obtained from these results. However, the formalization given in this paper is different, and closer to the original definition of stable models [Gelfond and Lifschitz, 1991], thus allowing to cover stability conditions based on subset-minimality rather than operator fixpoints. This is the case of F-stable models [Faber *et al.*, 2011; Ferraris, 2005], for which a characterization using approximation fixpoint theory appears to be difficult.

This work is also related to ASP systems [Gebser *et al.*, 2009; Leone *et al.*, 2006], where rule bodies clearly define Boolean functions. Common constructs in ASP are negation as failure and aggregate atoms. Other semantically similar constructs are DL [Eiter *et al.*, 2008] and HEX atoms [Eiter *et al.*, 2014], which extend ASP by allowing to interact with external knowledge bases, possibly expressed in different languages. All of these constructs fit in the framework considered in this paper. For example, the interpretation of negated literals in program reducts is fixed by many semantics [Pelov *et al.*, 2007; Son and Pontelli, 2007; Shen and Wang, 2012; Gelfond and Zhang, 2014; Ferraris, 2005] but not in [Faber *et al.*, 2011]. Negation as failure must be also taken into account in the definition of stable models by Gelfond and Zhang, whose program reducts directly refer Boolean function domains, and therefore do not allow to consider each rule body as a unique Boolean function. By constraining rule bodies of LPBFs to comprise exactly one, indivisible Boolean function, all these technical differences disappear, and properties relating to the interpretation of Boolean functions emerge. Indeed, Definitions 6–9 provide a uniform point of view of four different notions of stable models introduced in the literature, and clarify that they differ only minimally on program reducts.

8 Conclusion

Abstract dialectical frameworks and logic programs with Boolean functions are linked here by means of definitorial logic programs with Boolean functions, a framework on which several notions of stable models can be restated uniformly. Similarities and differences of the considered stable model semantics easily emerge from the new characterizations, which are focused on Boolean functions and are not distracted by other common constructs such as negation as failure. This work can mark the beginning of a potentially very fruitful research field that crosses over two research areas. Undoubtedly many more properties can be taken over from either field to the other, and also differences might surface. And, pragmatically perhaps most important, implemented systems can be used profitably for both ADFs and LPBFs. As an example, the ASP encoding mentioned in Section 4 for computing S-stable models of ADFs can be adapted to compute G-stable models of LPBFs.

References

[Alviano and Faber, 2013] Mario Alviano and Wolfgang Faber. Properties of answer set programming with convex generalized atoms. In Michael Fink and Yuliya Lierler, editors, *Sixth International Workshop on Answer Set Programming and Other Computing Paradigms (ASPOCP 2013)*, pages 3–16, 2013.

[Alviano *et al.*, 2011] Mario Alviano, Wolfgang Faber, Nicola Leone, Simona Perri, Gerald Pfeifer, and Giorgio Terracina. The disjunctive datalog system DLV. In Georg Gottlob, editor, *Datalog 2.0*, volume 6702 of *LNCS*, pages 282–301. Springer Berlin/Heidelberg, 2011.

[Alviano *et al.*, 2013] Mario Alviano, Carmine Dodaro, Wolfgang Faber, Nicola Leone, and Francesco Ricca. WASP: A native ASP solver based on constraint learning. In Pedro Cabalar and Tran Cao Son, editors, *Logic Programming and Nonmonotonic Reasoning, 12th International Conference, LPNMR 2013, Corunna, Spain, September 15-19, 2013. Proceedings*, volume 8148 of *LNCS*, pages 54–66. Springer, 2013.

[Alviano *et al.*, 2014] Mario Alviano, Carmine Dodaro, and Francesco Ricca. Anytime computation of cautious consequences in answer set programming. *TPLP*, 14(4-5):755–770, 2014.

[Brewka and Woltran, 2010] Gerhard Brewka and Stefan Woltran. Abstract dialectical frameworks. In Fangzhen Lin, Ulrike Sattler, and Mirosław Truszczyński, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Twelfth International Conference, KR 2010, Toronto, Ontario, Canada, May 9-13, 2010*, pages 102–111. AAAI Press, 2010.

[Brewka *et al.*, 2013] Gerhard Brewka, Hannes Strass, Stefan Ellmauthaler, Johannes Peter Wallner, and Stefan Woltran. Abstract dialectical frameworks revisited. In Francesca Rossi, editor, *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*, pages 803–809, 2013.

[Denecker *et al.*, 2004] Marc Denecker, Victor W. Marek, and Mirosław Truszczyński. Ultimate approximation and its application in nonmonotonic knowledge representation systems. *Inf. Comput.*, 192(1):84–121, 2004.

[Dung, 1995] Phan Minh Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artif. Intell.*, 77(2):321–358, 1995.

[Eiter *et al.*, 2008] Thomas Eiter, Giovambattista Ianni, Thomas Lukasiewicz, Roman Schindlauer, and Hans Tompits. Combining answer set programming with description logics for the semantic web. *Artif. Intell.*, 172(12-13):1495–1539, 2008.

[Eiter *et al.*, 2014] Thomas Eiter, Michael Fink, Thomas Krennwallner, Christoph Redl, and Peter Schüller. Efficient hex-program evaluation based on unfounded sets. *J. Artif. Intell. Res. (JAIR)*, 49:269–321, 2014.

[Ellmauthaler and Strass, 2014] Stefan Ellmauthaler and Hannes Strass. The DIAMOND system for computing with abstract dialectical frameworks. In Simon Parsons, Nir Oren, Chris Reed, and Federico Cerutti, editors, *Computational Models of Argument - Proceedings of COMMA 2014, Atholl Palace Hotel, Scottish Highlands, UK, September 9-12, 2014*, volume 266 of *Frontiers in Artificial Intelligence and Applications*, pages 233–240. IOS Press, 2014.

[Faber *et al.*, 2008] Wolfgang Faber, Gerald Pfeifer, Nicola Leone, Tina Dell'Armi, and Giuseppe Ielpa. Design and implementation of aggregate functions in the dlv system. *TPLP*, 8(5–6):545–580, 2008.

[Faber *et al.*, 2011] Wolfgang Faber, Nicola Leone, and Gerald Pfeifer. Semantics and complexity of recursive aggregates in answer set programming. *Artificial Intelligence*, 175(1):278–298, 2011. Special Issue: John McCarthy's Legacy.

[Ferraris, 2005] Paolo Ferraris. Answer Sets for Propositional Theories. In Chitta Baral, Gianluigi Greco, Nicola Leone, and Giorgio Terracina, editors, *Logic Programming and Nonmonotonic Reasoning — 8th International Conference, LPNMR'05, Diamante, Italy, September 2005, Proceedings*, volume 3662 of *LNCS*, pages 119–131. Springer Verlag, September 2005.

[Gebser *et al.*, 2009] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. On the implementation of weight constraint rules in conflict-driven ASP solvers. In Patricia M. Hill and David Scott Warren, editors, *Logic Programming, 25th International Conference, ICLP 2009, Pasadena, CA, USA, July 14-17, 2009. Proceedings*, volume 5649, pages 250–264. Springer, 2009.

[Gelfond and Lifschitz, 1988] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In Robert A. Kowalski and Kenneth A. Bowen, editors, *Logic Programming, Proceedings of the Fifth International Conference and Symposium*, pages 1070–1080. MIT Press, 1988.

[Gelfond and Lifschitz, 1991] Michael Gelfond and Vladimir Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Comput.*, 9(3/4):365–386, 1991.

[Gelfond and Zhang, 2014] Michael Gelfond and Yuanlin Zhang. Vicious circle principle and logic programs with aggregates. *TPLP*, 14(4-5):587–601, 2014.

[Leone *et al.*, 2006] Nicola Leone, Gerald Pfeifer, Wolfgang Faber, Thomas Eiter, Georg Gottlob, Simona Perri, and Francesco Scarcello. The DLV system for knowledge representation and reasoning. *ACM Trans. Comput. Log.*, 7(3):499–562, 2006.

[Pelov *et al.*, 2007] Nikolay Pelov, Marc Denecker, and Maurice Bruynooghe. Well-founded and Stable Semantics of Logic Programs with Aggregates. *TPLP*, 7(3):301–353, 2007.

[Shen and Wang, 2012] Yi-Dong Shen and Kewen Wang. FLP semantics without circular justifications for general logic programs. In Jörg Hoffmann and Bart Selman, editors, *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada.*, pages 576–591. AAAI Press, 2012.

[Son and Pontelli, 2007] Tran Cao Son and Enrico Pontelli. A Constructive Semantic Characterization of Aggregates in ASP. *TPLP*, 7:355–375, May 2007.

[Son *et al.*, 2007] Tran Cao Son, Enrico Pontelli, and Phan Huy Tu. Answer sets for logic programs with arbitrary abstract constraint atoms. *J. Artif. Intell. Res. (JAIR)*, 29:353–389, 2007.

[Strass and Wallner, 2014] Hannes Strass and Johannes Peter Wallner. Analyzing the computational complexity of abstract dialectical frameworks via approximation fixpoint theory. In Chitta Baral, Giuseppe De Giacomo, and Thomas Eiter, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Fourteenth International Conference, KR 2014, Vienna, Austria, July 20-24, 2014*, pages 101–110, 2014.

[Strass, 2013] Hannes Strass. Approximating operators and semantics for abstract dialectical frameworks. *Artif. Intell.*, 205:39–70, 2013.