On the Semantics of Recursive Bipolar AFs and Partial Stable Models*

Gianvincenzo Alfano, Sergio Greco, Francesco Parisi, and Irina Trubitsyna

DIMES Department, University of Calabria, Rende, Italy
{g.alfano,greco,fparisi,i.trubitsyna}@dimes.unical.it

Abstract. Extensions of Dung's abstract Argumentation Framework (AF) include the general class of Recursive Bipolar AFs (Rec-BAFs), i.e., AFs with recursive attacks and supports. Although the relationships between AF semantics and Partial Stable Models (PSMs) of logic programs has been deeply investigated, this is not the case for Rec-BAFs. In this paper we explore this relationship, showing that a Rec-BAF Δ can be translated into a logic program P_Δ so that the extensions of Δ under different argumentation semantics coincide with subsets of the PSMs of P_Δ. We provide a logic programming approach that characterizes, in an elegant and uniform way, the semantics of several AF-based frameworks which belong to the class of Rec-BAFs. This result allows also to define the semantics for new AF-based frameworks, such as AFs with recursive attacks and recursive deductive supports.

Keywords: Abstract Argumentation · Partial Stable Models · Semantics.

1 Introduction

Formal argumentation has emerged as one of the important fields in Artificial Intelligence [10, 36]. In particular, Dung's abstract Argumentation Framework (AF) is a simple, yet powerful formalism for modelling disputes between two or more agents [22]. An AF consists of a set of *arguments* and a binary *attack* relation over the set of arguments that specifies the *interactions* between arguments: intuitively, if argument a attacks argument b, then b is acceptable only if a is not. Hence, arguments are abstract entities whose role is entirely determined by the interactions specified by attacks.

Dung's framework has been extended in many different ways, including the introduction of new kinds of interactions between arguments and/or attacks. In particular, the class of *Bipolar Argumentation Frameworks (BAFs)* is an interesting extension of the AF which allows for also modelling the *support* between arguments [32, 37]. Different interpretations of the notion of support have been proposed [15, 18]. *Deductive* support [37] is intended to capture the following intuition: if argument a supports argument b, then the acceptance of a implies the acceptance of b; thus, the non-acceptance of b implies the non-acceptance of a. On the other hand, *necessary* support [32, 9] is interpreted in a dual way [15]: if a supports b, then the acceptance of a is necessary to

get the acceptance of b; equivalently, accepting b implies accepting a. An *AFN (AF with Necessities)* is a BAF where supports are interpreted as necessities [32]; an *AFD (AF with Deductive supports)* is a BAF where supports are interpreted as deductions [37]. Clearly, the way the support is interpreted changes the set of *extensions*, i.e. the set of acceptable elements of an argumentation framework.

Further extensions of the Dung framework consider second-order interactions [37], e.g., attacks to attacks/supports, as well as more general forms of interactions such as recursive AFs where attacks can be recursively attacked [9, 16] and recursive BAFs, where attacks/supports can be recursively attacked/supported [27, 17].

Following Dung's approach, the meaning of recursive AF-based frameworks is still given by relying on the concept of extension. However, the extensions of an *AF with Recursive Attacks (AFRA)* [9] and of an *Attack-Support Argumentation Framework (ASAF)* [19, 27] also include the (names of) attacks and supports that intuitively contribute to determine the set of accepted arguments. Particularly, the acceptability of an attack is related to the acceptability of its source argument: an attack in the AFRA is defeated even when its source argument is defeated. This is not the case for *Recursive AF (RAF)* [16] and *Recursive AF with Necessities (RAFN)* frameworks [17], which offer a different semantics for recursive AFs and recursive BAFs with necessary supports, respectively. From a syntax standpoint all the argumentation frameworks mentioned above can be viewed as (possibly restricted forms of) *Recursive Bipolar Argumentation Frameworks (Rec-BAFs)*, though semantically different because having different interpretations of support and different ways of determining the status of attacks.

Recently there has been an increasing interest in studying the relationships between argumentation frameworks and logic programming (LP). In particular, the semantic equivalence between complete extensions in AF and 3-valued stable models in LP was first established in [38]. Then, the relationships of LP with AF have been further studied in [13], whereas those with Assumption-Based Argumentation [12, 20] have been considered in [14], and those with Abstract Dialectical Frameworks have been investigated in [1]. Efficient mappings from AF to *Answer Set Programming* (i.e. LP with *Stable Model* semantics [26]) have been investigated as well [35, 24]. The well-know AF system ASPARTIX is implemented by rewriting the input AF into an ASP program and using an ASP solver to compute extensions. Although the ASPARTIX system allows also to reason on some extensions of AF, such as *Extended AF (EAF)* [31] and AFRA, so far the relationships between LP and more general frameworks extending AF such as Rec-BAFs has not been adequately studied. Thus, in this paper, we investigate these relationships by generalizing the work in [13] and providing relationships between LP and different recently proposed generalizations of the Dung's framework.

Contributions. The main contributions of this paper are as follows:

- We introduce a general approach for characterizing the extensions of different AF-based frameworks (e.g. AFRA, RAF, ASAF, RAFN) under several well-known argumentation semantics in terms of Partial Stable Models (PSMs) of logic programs. This is achieved by providing a modular definition of the sets of *defeated* and *acceptable* elements (i.e., arguments, attacks and supports) for each AF-based framework, and by leveraging on the connection between argumentation semantics and subsets of PSMs. In particular, for any argumentation framework belonging to

the class of Rec-BAFs, our formulation of acceptable elements allow us to easily and systematically derive a propositional program whose PSMs corresponds to the extensions (under a given semantics) of the given framework.
- Our approach is used to define new semantics for AFs with recursive attacks and supports under deductive interpretation of supports, where the status of an attack is considered independently from the status of its source. We call these frameworks *Recursive Argumentation Framework with Deductive supports (RAFD)* and *Argumentation Framework with Recursive Attacks and Deductive supports (AFRAD)*. It turns out that AFDs are special cases of these frameworks.

Our results can be used $i)$ for better understanding the semantics of several AF-based frameworks, $ii)$ to easily define new semantics for extended frameworks, and $iii)$ to provide additional tools for computing stable semantics using answer set solvers [25] and even other complete-based semantics using classical program rewriting [30] (see also [35, 24]).

An extended version of this paper can be found in [6]. Proofs are available in [7].

2 Preliminaries

We first recall Dung's framework and then introduce the general class of recursive bipolar argumentation frameworks.

2.1 Argumentation Frameworks

An abstract *Argumentation Framework* (AF) is a pair $\langle A, \Omega \rangle$, where A is a set of *arguments* and $\Omega \subseteq A \times A$ is a set of *attacks*. An AF can be seen as a directed graph, whose nodes represent arguments and edges represent attacks; an attack $(a, b) \in \Omega$ from a to b is represented by $a \rightarrow b$.

Different semantics notions have been defined leading to the characterization of collectively acceptable sets of arguments, called *extensions* [22]. Given an AF $\Delta = \langle A, \Omega \rangle$ and a set $\mathbf{S} \subseteq A$ of arguments, an argument $a \in A$ is said to be *i*) *defeated* w.r.t. $\mathbf{S}$ iff $\exists b \in \mathbf{S}$ such that $(b, a) \in \Omega$, and *ii*) *acceptable* w.r.t. $\mathbf{S}$ iff for every argument $b \in A$ with $(b, a) \in \Omega$, there is $c \in \mathbf{S}$ such that $(c, b) \in \Omega$. The sets of defeated and acceptable arguments w.r.t. $\mathbf{S}$ are defined as follows (where Δ is understood):

- $Def(\mathbf{S}) = \{a \in A \mid \exists\, b \in \mathbf{S}\,.\,(b, a) \in \Omega\}$;
- $Acc(\mathbf{S}) = \{a \in A \mid \forall\, b \in A\,.\,(b, a) \in \Omega \;\Rightarrow\; b \in Def(\mathbf{S})\}$.

Given an AF $\langle A, \Omega \rangle$, a set $\mathbf{S} \subseteq A$ of arguments is said to be $i)$ *conflict-free* iff $\mathbf{S} \cap Def(\mathbf{S}) = \emptyset$, and $ii)$ *admissible* iff it is conflict-free and $\mathbf{S} \subseteq Acc(\mathbf{S})$.

Given an AF $\langle A, \Omega \rangle$, a set $\mathbf{S} \subseteq A$ is an *extension* called:

- *complete* iff it is conflict-free and $\mathbf{S} = Acc(\mathbf{S})$;
- *preferred* iff it is a maximal (w.r.t. $\subseteq$) complete extension;
- *stable* iff it is a total preferred extension, i.e. a preferred extension s.t. $\mathbf{S} \cup Def(\mathbf{S}) = A$;

– *semi-stable* iff it is a preferred extension such that $\mathbf{S} \cup Def(\mathbf{S})$ is maximal;
– *grounded* iff it is the smallest (w.r.t. $\subseteq$) complete extension;
– *ideal* iff it is the biggest (w.r.t. $\subseteq$) complete extension contained in every preferred extension.

The set of complete (resp., preferred, stable, semi-stable, grounded, ideal) extensions of a framework Δ will be denoted by $\mathcal{CO}(\Delta)$ (resp., $\mathcal{PR}(\Delta)$, $\mathcal{ST}(\Delta)$, $\mathcal{SST}(\Delta)$, $\mathcal{GR}(\Delta)$, $\mathcal{ID}(\Delta)$).

Example 1. Let $\Delta = \langle A, \Omega \rangle$ be an AF where $A = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}$ and $\Omega = \{(\mathtt{a}, \mathtt{b}), (\mathtt{b}, \mathtt{a}), (\mathtt{a}, \mathtt{c}), (\mathtt{b}, \mathtt{c}), (\mathtt{c}, \mathtt{d}), (\mathtt{d}, \mathtt{c})\}$. The set of complete extension is $\mathcal{CO}(\Delta) = \{\emptyset, \{\mathtt{d}\}, \{\mathtt{a}, \mathtt{d}\}, \{\mathtt{b}, \mathtt{d}\}\}$. Consequently, $\mathcal{PR}(\Delta) = \mathcal{ST}(\Delta) = \mathcal{SST}(\Delta) = \{\{\mathtt{a}, \mathtt{d}\}, \{\mathtt{b}, \mathtt{d}\}\}$, $\mathcal{GR}(\Delta) = \{\emptyset\}$, $\mathcal{ID}(\Delta) = \{\{\mathtt{d}\}\}$. □

2.2 Recursive Bipolar Argumentation Frameworks

A *Recursive Bipolar Argumentation Framework (Rec-BAF)* is a tuple $\langle A, \Sigma, \Pi, \mathbf{s}, \mathbf{t} \rangle$, where A is a set of arguments, Σ is a set of attack names, Π is a set of necessary support names, $\mathbf{s}$ (resp., $\mathbf{t}$) is a function from $\Sigma \cup \Pi$ to A (resp., to $A \cup \Sigma \cup \Pi$) mapping each attack/support to its source (resp., target). In the following, given a set Φ such that either $\Phi \subseteq \Sigma$ or $\Phi \subseteq \Pi$, we denote by $i)$ $\Phi^* = \{(\mathbf{s}(\gamma), \mathbf{t}(\gamma)) \mid \gamma \in \Phi\}$ the set of pairs connected by an attack/support edge, and $ii)$ Φ^+ the transitive closure of Φ. It is assumed that Π^* is acyclic.

Two different semantics have been defined under necessary interpretation of supports, as recalled in what follows.

Recursive AF with Necessities. The *Recursive Argumentation Framework with Necessities (RAFN)* has been proposed in [17]. The semantics combines the RAF interpretation of attacks [16] with that of BAF under the necessity interpretation of supports (i.e., AFN) [32]. Here we consider a simplified version where supports have a single source and the support relation is acyclic. Formally, given an RAFN $\langle A, \Sigma, \Pi, \mathbf{s}, \mathbf{t} \rangle$, $X \in (A \cup \Sigma \cup \Pi)$, $a \in A$, and $\mathbf{S} \subseteq A \cup \Sigma \cup \Pi$, we say that argument a *recursively attacks* X *given* $\mathbf{S}$ (denoted as a $\mathtt{att}_{\mathbf{S}}$ X) if either $(a, X) \in (\Sigma \cap \mathbf{S})^*$ or there exists $b \in A$ such that $(a, b) \in (\Sigma \cap \mathbf{S})^*$ and $(b, X) \in (\Pi \cap \mathbf{S})^+$.

For any RAFN Δ and $\mathbf{S} \subseteq A \cup \Sigma \cup \Pi$, the *defeated* and *acceptable* sets (given $\mathbf{S}$) are defined as follows:

– $Def(\mathbf{S}) = \{X \in A \cup \Sigma \cup \Pi \mid \exists b \in A \cap \mathbf{S} \,.\, b \; \mathtt{att}_{\mathbf{S}} \; X\}$;
– $Acc(\mathbf{S}) = \{X \in A \cup \Sigma \cup \Pi \mid \forall b \in A \,.\, b \; \mathtt{att}_{\mathbf{S}} \; X \Rightarrow b \in Def(\mathbf{S})\}$.

Attack-Support AF. The *Attack-Support Argumentation Framework (ASAF)* has been proposed in [19, 27]. The semantics combines the AFRA interpretation of attacks [9] with that of BAF under the necessary interpretation of supports (i.e., AFN). For the sake of presentation, we consider a slight generalization of ASAF, where attack and support names are first-class citizens, giving the possibility to represent multiple attacks and supports from the same source to the same target.[1]

[1] In the original work [19, 27] an ASAF is a tuple $\langle A, \Omega, \Gamma \rangle$ where A is a set of arguments, Ω is a set of attacks $\Omega : A \to (A \cup \Omega)$, and Γ is a set of supports $\Gamma : A \to (A \cup \Gamma)$.

Formally, given an ASAF $\langle A, \Sigma, \Pi, \mathbf{s}, \mathbf{t}\rangle$, $X \in (A \cup \Sigma \cup \Pi)$, $\alpha \in \Sigma$, and $\mathbf{S} \subseteq A \cup \Sigma \cup \Pi$, we say that i) α *(directly or indirectly) attacks* X (denoted by α $\mathtt{def}$ X) if either $\mathbf{t}(\alpha) = X$ or $\mathbf{t}(\alpha) = \mathbf{s}(X)$, and ii) α *extendedly defeats* X *given* $\mathbf{S}$ (denoted as α $\mathtt{def}_{\mathbf{S}}$ X) if either α $\mathtt{def}$ X or there exists $b \in A$ such that $\mathbf{t}(\alpha) = b$ and either $(b, X) \in (\Pi \cap \mathbf{S})^+$ or $(b, \mathbf{s}(X)) \in (\Pi \cap \mathbf{S})^+$. For any ASAF Δ and $\mathbf{S} \subseteq A \cup \Sigma \cup \Pi$, the *defeated* and *acceptable* sets (given $\mathbf{S}$) are:

- $Def(\mathbf{S}) = \{X \in A \cup \Sigma \cup \Pi \mid \exists\, \alpha \in \Sigma \cap \mathbf{S} \,.\, \alpha\ \mathtt{def}_{\mathbf{S}}\ X\}$;
- $Acc(\mathbf{S}) = \{X \in A \cup \Sigma \cup \Pi \mid \forall \alpha \in \Sigma \,.\, \alpha\ \mathtt{def}_{\mathbf{S}}\ X \Rightarrow \alpha \in Def(\mathbf{S})\}$.

The notions of *conflict-free*, *admissible sets*, and the different types of extensions can be defined in a standard way (see Section 2.1) by considering $\mathbf{S} \subseteq A \cup \Sigma \cup \Pi$ and by using the definitions of defeated and acceptable sets reported above.

AFs with high-order interactions can be mapped to AFs, though the mapping is not trivial because extensions also contain attacks and supports. In particular, an equivalent AF for an ASAF can be obtained by translating it into an AFN [19] that in turns can be translated into an AF [32] (see also [27]).

Example 2. Consider a situation where we want to decide whether to play tennis on the basis of some information. Assume we have the following arguments: `r` (it is raining), `p` (play tennis), `o` (the tennis court is outside), and the logical implications: (α_1) if it is raining, then we do not play tennis, and (β_1) if the tennis court is outside supports, then implication α_1 should hold. This situation can be modelled using the Rec-BAF Δ shown in Figure 1, where α_1 is an *attack* (denoted by $\rightarrow$), and β_1 is a *support* (denoted by $\Rightarrow$). Under both ASAF and RAFN semantics $\mathcal{CO}(\Delta) = \{\{\mathtt{o}, \mathtt{r}, \alpha_1, \beta_1\}\}$. □

AF-based frameworks belonging to the class of Rec-BAFs ASAFs and RAFNs generalize other AF-based frameworks mentioned in the Introduction. In particular, (*i*) an AFRA is an ASAF $\langle A, \Sigma, \Pi, \mathbf{s}, \mathbf{t}\rangle$ where $\Pi = \emptyset$ [2]; (*ii*) an RAF is an RAFN $\langle A, \Sigma, \Pi, \mathbf{s}, \mathbf{t}\rangle$ where $\Pi = \emptyset$; and (*iii*) we can think of an AFN as either an ASAF or an RAFN $\langle A, \Sigma, \Pi, \mathbf{s}, \mathbf{t}\rangle$ where function $\mathbf{t} : \Sigma \cup \Pi \rightarrow A$ maps attacks and supports to arguments only.

It is important to observe that different frameworks extending AF share the same structure, although they have different semantics. Thus we distinguish between framework and *class* of frameworks. Two frameworks sharing the same syntax (i.e. the structure) belong to the same (syntactic) class. For instance, BAF is a syntactic class, whereas AFN and AFD are two specific frameworks sharing the same BAF syntax; their semantics differ because they interpret supports in different ways. *Recursive AF (Rec-AF)* is another syntactic class, where AFs are extended by allowing recursive attacks: AFRA and RAF are frameworks belonging to this class, which differ only in the determination of the status of attacks. Finally, ASAF and RAFN are two frameworks belonging

[2] For the sake of presentation, we consider a slight generalization of AFRA, where attack names are first-class citizens, allowing to also represent more than one attack from the same source to the same target. In the original work an AFRA is a tuple $\langle A, \Omega\rangle$ where A is a set of arguments and Ω is a set of attacks $\Omega : A \rightarrow (A \cup \Omega)$ [9].

Acronym	Framework
AF	abstract Argumentation Framework [Dung,1995]
BAF	Bipolar AF
AFN	AF with Necessities [Nouia and Risch,2011]
AFD	AF with Deductive supports [Villata et al.,2012]
Rec-AF	Recursive-AF
AFRA	AF with Recursive Attacks [Baroni et al.,2011]
RAF	Recursive AF [Cayrol et al.,2017]
Rec-BAF	Recursive-BAF
ASAF	Attack-Support AF [Gottifredi et al.,2018]
RAFN	Recursive AF with Necessities [Cayrol et al.,2018]
AFRAD	AF with Rec. Attacks and Deductive supports
RAFD	Recursive AF with Deductive supports

Table 1: AF-based frameworks considered in the paper. Syntactic classes are in bold.

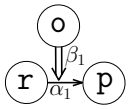

Fig. 1: Rec-BAF of Example 2

to the general class Rec-BAF presented in this section, consisting in the extension of BAF with recursive attacks and supports. The differences between ASAF and RAFN semantics are not in the way they interpret supports (both based on the necessity interpretation), but in a different determination of the status of attacks as they extend AFRA and RAF, respectively. In Section 3.2 we will present two new frameworks that also belong to the Rec-BAF class: *Recursive Argumentation Framework with Deductive supports (RAFD)* and *Argumentation Framework with Recursive Attacks and Deductive supports (AFRAD)*. AFDs are special cases of these frameworks.

Thus, the class of Rec-BAF includes the sub-classes BAF and Rec-AF, and thus the 9 frameworks discussed above, ASAF, RAFN, AFRAD, and RAFD, and their specializations AF, AFRA, RAF, AFN, and AFD. We use $\mathfrak{F}$ to denote the set of these 9 frameworks. Table 1 lists the frameworks in $\mathfrak{F}$. With a little abuse of notation, we will use the same symbol Δ to denote any framework in $\mathfrak{F}$.

2.3 Partial Stable Models

In this section, we review the basic concepts which underly the notion of PSMs [34].

A (normal, logic) program is a set of rules of the form $A \leftarrow B_1 \wedge \cdots \wedge B_n$, with $n \geq 0$, where A is an atom, called head, and $B_1 \wedge \cdots \wedge B_n$ is a conjunction of literals, called body. We consider programs without function symbols. Given a program P, $ground(P)$ denotes the set of all ground instances of the rules in P. The Herbrand Base of a program P, i.e. the set of all ground atoms which can be constructed using predicate and constant symbols occurring in P, is denoted by B_P, whereas $\neg B_P$ denotes the set $\{\neg A \mid A \in B_P\}$. Analogously, for any set $S \subseteq B_P \cup \neg B_P$, $\neg S$ denotes the set $\{\neg A \mid A \in S\}$, where $\neg\neg A = A$. Given $I \subseteq B_P \cup \neg B_P$, $pos(I)$ (resp., $neg(I)$) stands for $I \cap B_P$ (resp., $\neg I \cap B_P$). I is *consistent* if $pos(I) \cap \neg neg(I) = \emptyset$, otherwise I is *inconsistent*.

Given a program P, $I \subseteq B_P \cup \neg B_P$ is an *interpretation* of P if I is consistent. Also, I is *total* if $pos(I) \cup neg(I) = B_P$, *partial* otherwise. A partial interpretation M of a

program P is a *partial model* of P if for each $\neg A \in M$ every rule in $ground(P)$ having as head A contains at least one body literal B such that $\neg B \in M$. Given a program P and a partial model M, the positive instantiation of P w.r.t. M, denoted by P^M, is obtained from $ground(P)$ by deleting: (a) each rule containing a negative literal $\neg A$ such that $A \in pos(M)$; (b) each rule containing a literal B such that neither B nor $\neg B$ is in M; (c) all the negative literals in the remaining rules. Clearly, all the rules in P are definite clauses and hence the minimal Herbrand model of P can be obtained as the least fixpoint of its immediate consequence operator T_{P^M}, denoted by $T^{\omega}_{P^M}(\emptyset)$. For any partial model M of a logic program P, $T^{\omega}_{P^M}(\emptyset) \subseteq M$ [34].

Let P be a program and M a partial model for P. Then M is (a) *founded* if $T^{\omega}_{P^M}(\emptyset) = pos(M)$; (b) *stable* if it is founded and it is not a proper subset of any other founded model. The set of partial stable models of a logic program P, denoted by $\mathcal{PM}(P)$, define a meet semi-lattice. The *well-founded* model (denoted by $\mathcal{WF}(P)$) and the *maximal-stable* models $\mathcal{MS}(P)$[3], are defined by considering $\subseteq$-minimal and $\subseteq$-maximal elements. The set of (total) *stable* models (denoted by $\mathcal{SM}(P)$) is obtained by considering the maximal-stable models which are total, whereas the *least-undefined* models (denoted by $\mathcal{LM}(P)$) are obtained by considering the maximal-stable models with a $\subseteq$-minimal set of undefined atoms (i.e., atoms which are neither true or false). The *max-deterministic* model (denoted by $\mathcal{MD}(P)$) is the $\subseteq$-maximal PSM contained in every maximal-stable model [33, 29].

Example 3. Consider the program P consisting of the following four rules $\{\mathtt{a} \leftarrow \neg\mathtt{b};\ \mathtt{b} \leftarrow \neg\mathtt{a};\ \mathtt{c} \leftarrow \neg\mathtt{a} \wedge \neg\mathtt{b} \wedge \neg\mathtt{d};\ \mathtt{d} \leftarrow \neg\mathtt{c}\}$. The set of partial stable models of P is $\mathcal{PS}(P) = \{\ \emptyset,\ \{\neg\mathtt{c}, \mathtt{d}\}, \{\mathtt{a}, \neg\mathtt{b}, \neg\mathtt{c}, \mathtt{d}\},\ \{\neg\mathtt{a}, \mathtt{b}, \neg\mathtt{c}, \mathtt{d}\}\ \}$. Consequently, $\mathcal{WF}(\Delta) = \{\ \emptyset\ \}$, $\mathcal{MD}(\Delta) = \{\ \{\neg\mathtt{c}, \mathtt{d}\}\ \}$, $\mathcal{MS} = \mathcal{ST}(\Delta) = \mathcal{LS}(\Delta) = \{\{\mathtt{a}, \neg\mathtt{b}, \neg\mathtt{c}, \mathtt{d}\}, \{\neg\mathtt{a}, \mathtt{b}, \neg\mathtt{c}, \mathtt{d}\}\}$. □

Propositional Programs. Given a set of symbols $\Lambda = \{a_1, ..., a_n\}$, a *(propositional) program* over Λ is a set of $|\Lambda|$ rules $a_i \leftarrow body_i$ $(1 \leq i \leq n)$, where every $body_i$ is a propositional formula defined over Λ. The semantics of a propositional program P, defined over a given alphabet Λ, is given in terms of the set $\mathcal{PS}(P)$ of its Partial Stable Models (PSMs) that are obtained as follows: *i*) P is first rewritten into a set of standard (ground) logic rules P', whose bodies contain conjunction of literals (even by adding fresh symbols to the alphabet)[4]; *ii*) next, the set of PSMs of P' is computed; *iii*) finally, fresh literals added to Λ in the first step are deleted from the models. It is worth noting that for propositional programs we can assume as Herbrand Base the set of (ground) atoms occurring in the program.

3 AF-based Frameworks and Partial Stable Models

In this section we present a new way to define the semantics of AF-based frameworks by considering propositional programs and partial stable models. In order to compare extensions E of a given framework Δ (containing acceptable elements) with PSMs of

[3] Corresponding to the *preferred extensions* of [21].

[4] A rule $a \leftarrow (b \vee c) \wedge (d \vee e)$ is rewritten as $a \leftarrow \neg a_1 \wedge \neg a_2,\ a_1 \leftarrow \neg b \wedge \neg c$ and $a_2 \leftarrow \neg d \wedge \neg e$.

a given program P (containing true and false atoms), we denote as $\widehat{E} = E \cup \{\neg a \mid a \in Def(E)\}$ the *completion* of E. Moreover, for a collection of extensions $\mathbf{E}$, $\widehat{\mathbf{E}}$ denotes the set $\{\widehat{E} \mid E \in \mathbf{E}\}$.

Observe also that for any framework Δ and complete extension E for Δ, elements not occurring in $E \cup Def(E)$ are said to be undecided (or undefined), whereas for any program P and PSM M for P, atoms not occurring in $pos(M) \cup neg(M)$ are said to be undefined. Thus, to compare complete extensions and PSMs it is sufficient to consider the completion of extensions.

The next proposition states the relationship between the argumentation frameworks (e.g. AF, BAF, Rec-BAF) and logic programs with partial stable models.

Proposition 1. *For any framework $\Delta \in \mathfrak{F}$ and propositional program P, whenever $\widehat{\mathcal{CO}(\Delta)} = \mathcal{PS}(P)$ it holds that $\widehat{\mathcal{PR}(\Delta)} = \mathcal{MS}(P)$, $\widehat{\mathcal{ST}(\Delta)} = \mathcal{ST}(P)$, $\widehat{\mathcal{SST}(\Delta)} = \mathcal{LM}(P)$, $\widehat{\mathcal{GR}(\Delta)} = \mathcal{WF}(P)$, and $\widehat{\mathcal{ID}(\Delta)} = \mathcal{MD}(P)$.*[5]

The result of Proposition 1 derives from the fact that preferred, stable, semi-stable, grounded, and ideal extensions are defined by selecting a subset of the complete extensions satisfying given criteria (see Section 2). On the other side, the maximal, stable, least-undefined, well-founded, and max-deterministic (partial) stable models are obtained by selecting a subset of the PSMs satisfying criteria coinciding with those used to restrict the set of complete extensions.

Given a framework Δ and an extension E, for any element a which could occur in some extension of Δ, the truth value $v_{_E}(a)$, or simply $v(a)$ whenever E is understood, is equal to `true` if $a \in E$, `false` if $a \in Def(E)$, `undec` (*undecided*) otherwise. Hereafter, we assume that $\texttt{false} < \texttt{undec} < \texttt{true}$ and $\neg\texttt{undec} = \texttt{undec}$.

The strict relationship between the semantics of AFs (given in terms of subset of complete extensions) and the semantics of logic programs (given in terms of subset of PSMs) has been shown in [38, 13]. The relationship is based on the observation that the meaning of an attack $a \rightarrow b$ is that the condition $v(b) \leq \neg v(a)$ must hold. On the other side, the satisfaction of a logical rule $a \leftarrow b_1, ..., b_n$ implies that $v(a) \geq min\{v(b_1), ..., v(b_n), \texttt{true}\}$.

Definition 1. *Given an AF $\Delta = \langle A, \Omega \rangle$, we denote as $P_\Delta = \{a \leftarrow \bigwedge_{(b,a)\in\Omega} \neg b \mid a \in A\}$ the propositional program derived from Δ.*

The semantics of an AF Δ can be obtained by considering PSMs of the logic program P_Δ. Particularly, for any AF Δ, $\widehat{\mathcal{CO}(\Delta)} = \mathcal{PS}(P_\Delta)$.

In the rest of this section, we show how the semantics defined for frameworks extending AF can be captured by means of PSMs of logic programs. We propose a general method that can be applied to all the discussed frameworks, and even to new frameworks in Section 3.2. Specifically, to model frameworks extending Dung's framework by logic programs under PSM semantics, we provide new definitions of defeated and acceptable sets that, for a given set $\mathbf{S}$, will be denoted by $\textsc{Def}(\mathbf{S})$ and $\textsc{Acc}(\mathbf{S})$, respectively. These definitions will be used to derive rules in P_Δ, the propositional program for $\Delta \in \mathfrak{F}$. For AFs we have that for every set $\mathbf{S} \subseteq A$, $\textsc{Def}(\mathbf{S}) = Def(\mathbf{S})$ and $\textsc{Acc}(\mathbf{S}) = Acc(\mathbf{S})$.

[5] For the novel frameworks $\Delta \in \{\text{AFRAD, RAFD}\}$, the set $\mathcal{CO}(\Delta)$ of the complete extensions, and the sets of extensions prescribed by the other semantics, are defined in Section 3.2.

3.1 Recursive BAFs with Necessary Supports

In this section we study the relationship between partial stable models and the semantics of Rec-BAFs. Particularly, we first present results for RAFN semantics, and then we discuss results for the ASAF framework. We remand to the next section the presentation of two novel semantics for recursive bipolar AFs with deductive interpretation of supports.

RAFN. We next provide the definitions of defeated and acceptable sets for an RAFN.

Definition 2. *For any RAFN $\langle A, \Sigma, \Pi, \mathbf{s}, \mathbf{t}\rangle$ and set $\mathbf{S} \subseteq A \cup \Sigma \cup \Pi$, we have that:*

- $\textsc{Def}(\mathbf{S}) = \{X \in A \cup \Sigma \cup \Pi \mid (\exists \alpha \in \Sigma \cap \mathbf{S} \,.\, \mathbf{s}(\alpha) \in \mathbf{S} \wedge \mathbf{t}(\alpha) = X) \ \vee$
  $(\exists\, \beta \ \in \ \Pi \cap \ \mathbf{S} \,.\, \mathbf{s}\,(\beta) \ \in \ \textsc{Def}(\mathbf{S}) \ \wedge \ \mathbf{t}(\beta) = X)\ \}$;
- $\textsc{Acc}(\mathbf{S}) = \{X \in A \cup \Sigma \cup \Pi \mid (\forall \alpha \in \Sigma \,.\, \mathbf{t}(\alpha) = X \Rightarrow (\alpha \in \textsc{Def}(\mathbf{S}) \vee$
  $\mathbf{s}(\alpha) \in \textsc{Def}(\mathbf{S}))) \wedge (\forall \beta \in \Pi \,.\, \mathbf{t}(\beta) = X \Rightarrow (\beta \in \textsc{Def}(\mathbf{S}) \vee \mathbf{s}(\beta) \in \textsc{Acc}(\mathbf{S})))\ \}$.

It is worth noting that $\textsc{Def}(\mathbf{S})$ and $\textsc{Acc}(\mathbf{S})$ are defined recursively, and whenever $\mathbf{S}$ is a complete extension we obtain the following result.

Theorem 1. *Given an RAFN Δ and an extension $\mathbf{S} \in \mathcal{CO}(\Delta)$, then $Def(\mathbf{S}) = \textsc{Def}(\mathbf{S})$ and $Acc(\mathbf{S}) = \textsc{Acc}(\mathbf{S})$.*

Theorem 1 states that in order to define the semantics for an RAFN Δ we can use acceptable sets $\mathbf{S} = \textsc{Acc}(\mathbf{S})$. This is captured by the following definition, that shows how to derive a propositional program from an RAFN from Definition 2.

Definition 3. *Given an RAFN $\Delta = \langle A, \Sigma, \Pi, \mathbf{s}, \mathbf{t}\rangle$, then P_Δ (the propositional program derived from Δ) contains, for each $X \in A \cup \Sigma \cup \Pi$, a rule*

$$X \leftarrow \bigwedge_{\alpha \in \Sigma \wedge \mathbf{t}(\alpha) = X} (\neg\alpha \vee \neg\mathbf{s}(\alpha)) \wedge \bigwedge_{\beta \in \Pi \wedge \mathbf{t}(\beta) = X} (\neg\beta \vee \mathbf{s}(\beta)).$$

Intuitively, starting from the definition of $\textsc{Acc}(\mathbf{S})$ in Definition 2, the rationale of the above definition is as follows. An element $X \in A \cup \Sigma \cup \Pi$ is `true` if (*i*) every attack α toward X is `false` or originates from a source $\mathbf{s}(\alpha)$ which is `false`, and (*ii*) every support β toward X is `false` or originates from a source $\mathbf{s}(\beta)$ which is `true`. These conditions resemble the conditions $(\forall \alpha \in \Sigma \,.\, \mathbf{t}(\alpha) = X \Rightarrow (\alpha \in \textsc{Def}(\mathbf{S}) \vee \mathbf{s}(\alpha) \in \textsc{Def}(\mathbf{S})))$ and $(\forall \beta \in \Pi \,.\, \mathbf{t}(\beta) = X \Rightarrow (\beta \in \textsc{Def}(\mathbf{S}) \vee \mathbf{s}(\beta) \in \textsc{Acc}(\mathbf{S})))$, respectively, of Definition 2 after interpreting elements in $\textsc{Def}(\mathbf{S})$ as `false` and elements in $\textsc{Acc}(\mathbf{S})$ as `true`.

As stated below, the set of complete extensions of an RAFN Δ coincides with the set of PSMs of P_Δ.

Theorem 2. *For any RAFN Δ, $\widehat{\mathcal{CO}(\Delta)} = \mathcal{PS}(P_\Delta)$.*

The previous theorem states that the set of complete extensions of an RAFN Δ coincides with the set of PSMs of the derived logic program P_Δ. Consequently, using Proposition 1, also the others argumentation semantics turns out to be characterized in

terms of subsets of PSMs. Moreover, previous results also apply to restricted frameworks such as RAF, where $\Pi = \emptyset$, and AFN, where $\mathbf{t} : \Sigma \cup \Pi \to A$.

ASAF. We next provide definitions of defeated and acceptable sets for an ASAF. They will be used in a way similar to that described above in order to obtain a semantically-equivalent propositional program for ASAFs.

Definition 4. *Given an ASAF $\langle A, \Sigma, \Pi, \mathbf{s}, \mathbf{t}\rangle$ and a set $\mathbf{S} \subseteq A \cup \Sigma \cup \Pi$, we define:*

- $\textsc{Def}(\mathbf{S}) = \{X \in A \cup \Sigma \cup \Pi \mid (X \in \Sigma \wedge \mathbf{s}(X) \in \textsc{Def}(\mathbf{S})) \vee (\exists \alpha \in \Sigma \cap \mathbf{S} \,.\, \mathbf{t}(\alpha) = X) \vee (\exists \beta \in \Pi \cap \mathbf{S} \,.\, \mathbf{t}(\beta) = X \wedge \mathbf{s}(\beta) \in \textsc{Def}(\mathbf{S}))\}$;
- $\textsc{Acc}(\mathbf{S}) = \{X \in A \cup \Sigma \cup \Pi \mid (X \in \Sigma \Rightarrow \mathbf{s}(X) \in \textsc{Acc}(\mathbf{S})) \wedge (\forall \alpha \in \Sigma \,.\, \mathbf{t}(\alpha) = X \Rightarrow \alpha \in \textsc{Def}(\mathbf{S})) \wedge (\forall \beta \in \Pi \,.\, \mathbf{t}(\beta) = X \Rightarrow (\beta \in \textsc{Def}(\mathbf{S}) \vee \mathbf{s}(\beta) \in \textsc{Acc}(\mathbf{S})))\}$.

Analogously to what has been done RAFNs, since it can be shown that for any complete extension $\mathbf{S}$ of an ASAF Δ it is the case that $Acc(\mathbf{S}) = \textsc{Acc}(\mathbf{S})$, the propositional program for an ASAF Δ can be derived by looking at the definition $\textsc{Acc}(\mathbf{S})$ of acceptable elements. In this case, the three conjuncts in the acceptance condition for an element X will correspond to three (group of) conjuncts, respectively, in the rule for X of the program P_Δ. Specifically, the last two conjuncts in the definition of $\textsc{Acc}(\mathbf{S})$ can be mapped by reasoning similarly to the RAFN case, whereas the first one entails a rule's body conjunction stating that if X is an attack then all of its sources must be `true`.

Definition 5. *For any ASAF $\Delta = \langle A, \Sigma, \Pi, \mathbf{s}, \mathbf{t}\rangle$, P_Δ (the propositional program derived from Δ) contains, for each $X \in A \cup \Sigma \cup \Pi$, a rule of the form*

$$X \leftarrow \varphi(X) \wedge \bigwedge_{\alpha \in \Sigma \wedge \mathbf{t}(\alpha) = X} \neg\alpha \wedge \bigwedge_{\beta \in \Pi \wedge \mathbf{t}(\beta) = X} (\neg\beta \vee \mathbf{s}(\beta))$$

where: $\varphi(X) = \mathbf{s}(X)$ if $X \in \Sigma$; otherwise, $\varphi(X) =$ `true` (recall that $\psi \wedge$ `true` $\equiv \psi$).

The set of complete extensions of an ASAF Δ coincides with the set of PSMs of the derived logic program P_Δ, meaning that using Proposition 1 also the others argumentation semantics turns out to be characterized in terms of subsets of PSMs.

Theorem 3. *For any ASAF Δ, $\widehat{\mathcal{CO}(\Delta)} = \mathcal{PS}(P_\Delta)$.*

Similarly to the case of RAFN, the above results also apply to restricted frameworks such as AFRA, where $\Pi = \emptyset$, and AFN, where $\mathbf{t} : \Sigma \cup \Pi \to A$.

Example 4. Consider a Rec-BAF Δ' obtained by adding to the Rec-BAF Δ of Example 2 argument `s` (it is sunny). Argument `r` and `s` attack each other through attacks α_2 and α_3, as shown in Figure 2. As shown below, Δ' has different extensions when viewed as an RAFN or an ASAF.

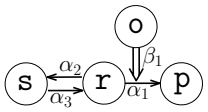

Fig. 2: Rec-BAF of Example 4

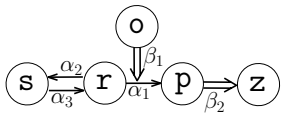

Fig. 3: Rec-BAF of Example 6

Under RAFN semantics Δ' has the following two complete extensions: $\{\mathtt{o}, \mathtt{s}, \mathtt{p}, \alpha_1, \alpha_2, \alpha_3, \beta_1\}$ and $\{\mathtt{o}, \mathtt{r}\ \alpha_1, \alpha_2, \alpha_3, \beta_1\}$. Differently, under ASAF semantics Δ' has the following complete extensions: $\{\mathtt{o}, \mathtt{s}, \mathtt{p},\ \alpha_3, \beta_1\}$ and $\{\mathtt{o}, \mathtt{r}\ \alpha_1, \alpha_2, \beta_1\}$. Note that attacks α_1 and α_2 (resp., α_3) do not belong to $\{\mathtt{o}, \mathtt{s}, \mathtt{p},\ \alpha_3, \beta_1\}$ since their source, $\mathtt{r}$, is defeated. Similarly, α_3 does not belong to $\{\mathtt{o}, \mathtt{r}\ \alpha_1, \alpha_2, \beta_1\}$ since its source, $\mathtt{s}$, is defeated. The propositional program under the RAFN semantics is the following:
$P_{\Delta'} = \{(\mathtt{o} \leftarrow), (\mathtt{r} \leftarrow \neg\alpha_3 \vee \neg\mathtt{s}), (\mathtt{p} \leftarrow \neg\alpha_1 \vee \neg\mathtt{r}), (\mathtt{s} \leftarrow \neg\alpha_2 \vee \neg\mathtt{r}), (\alpha_1 \leftarrow \neg\beta_1 \vee \mathtt{o}),$
$(\alpha_2 \leftarrow), (\alpha_3 \leftarrow), (\beta_1 \leftarrow)\}$, whose set of partial stable models is as follows:
$\mathcal{PS}(P_{\Delta'}) = \{M_1 = \{\mathtt{o}, \neg\mathtt{r}, \mathtt{s}, \mathtt{p}, \alpha_1, \alpha_2, \alpha_3, \beta_1\}, M_2 = \{\mathtt{o}, \mathtt{r}, \neg\mathtt{s}, \neg\mathtt{p}, \alpha_1, \alpha_2, \alpha_3, \beta_1\}\}$.

Analogously, the propositional program for Δ' under the ASAF semantics is:
$P_{\Delta'} = \{(\mathtt{o} \leftarrow), (\mathtt{r} \leftarrow \neg\alpha_3), (\mathtt{p} \leftarrow \neg\alpha_1), (\mathtt{s} \leftarrow \neg\alpha_2), (\alpha_1 \leftarrow \mathtt{r} \wedge (\neg\beta_1 \vee \mathtt{o})),$
$(\alpha_2 \leftarrow \mathtt{r}), (\alpha_3 \leftarrow \mathtt{s}), (\beta_1 \leftarrow)\}$, whose set of partial stable models is: $\mathcal{PS}(P_{\Delta'}) = \{M_3 = \{\mathtt{o}, \neg\mathtt{r}, \mathtt{s}, \mathtt{p}, \neg\alpha_1, \neg\alpha_2, \alpha_3, \beta_1\},\ M_4 = \{\mathtt{o}, \mathtt{r}, \neg\mathtt{s}, \neg\mathtt{p}, \alpha_1, \alpha_2, \neg\alpha_3, \beta_1\}\}$. It is worth noting that M_1 (resp., M_2) differs from M_3 (resp., M_4) in the status of α_1 and α_2 (resp., α_3). □

3.2 Recursive BAFs with Deductive Supports

In this section we study two new frameworks both belonging to the Rec-BAF class and both extending AFD by allowing recursive attacks and deductive supports. The first one, called *Recursive Argumentation Framework with Deductive supports (RAFD)*, extends RAF, whereas the second one, called *Argumentation Framework with Recursive Attacks and Deductive supports (AFRAD)*, extends AFRA. It is again assumed that Π^* is acyclic.

As we shall define the semantics by defining directly the sets $\textsc{Def}(\mathbf{S})$ and $\textsc{Acc}(\mathbf{S})$, differently from the previous section, we do not have any results regarding the equivalence between the sets $Acc(\mathbf{S})$ and $\textsc{Acc}(\mathbf{S})$ for $\mathbf{S} = \textsc{Acc}(\mathbf{S})$.

RAFD. As usual, we first define the sets of defeated and acceptable elements, and then the propositional logic program for an RAFD.

Definition 6. *For any RAFD $\langle A, \Sigma, \Pi, \mathbf{s}, \mathbf{t}\rangle$ and set $\mathbf{S} \subseteq A \cup \Sigma \cup \Pi$, we have that:*

- $\textsc{Def}(\mathbf{S}) = \{X \in A \cup \Sigma \cup \Pi \mid (\exists \alpha \in \Sigma \cap \mathbf{S} \,.\, \mathbf{t}(\alpha) = X \wedge \mathbf{s}(\alpha) \in \mathbf{S}) \ \vee$
  $(\exists \beta \in \Pi \cap \mathbf{S} \,.\, \mathbf{s}(\beta) = X \wedge \mathbf{t}\,(\beta) \in \textsc{Def}(\mathbf{S}))\ \}$;
- $\textsc{Acc}(\mathbf{S}) = \{X \in A \cup \Sigma \cup \Pi \mid (\forall \alpha \in \Sigma \,.\, \mathbf{t}(\alpha) = X \Rightarrow (\alpha \in \textsc{Def}(\mathbf{S}) \vee$
  $\mathbf{s}(\alpha) \in \textsc{Def}(\mathbf{S}))) \wedge (\forall \beta \in \Pi \,.\, \mathbf{s}(\beta) = X \ \Rightarrow (\beta \in \textsc{Def}(\mathbf{S}) \vee \mathbf{t}(\beta) \in \textsc{Acc}(\mathbf{S})))\}$.

The sets of extensions prescribed by the different semantics are based on the defeated and acceptable sets defined above. That is, given an RAFD $\Delta = \langle A, \Sigma, \Pi, \mathbf{s}, \mathbf{t}\rangle$, a set $\mathbf{S} \subseteq A \cup \Sigma \cup \Pi$ of elements is a complete extension of Δ iff it is conflict-free

(i.e., $\mathbf{S} \cap \textsc{Def}(\mathbf{S}) = \emptyset$) and $\mathbf{S} = \textsc{Acc}(\mathbf{S})$. As done for the other frameworks, we use $\mathcal{CO}(\Delta)$ to denote the set of complete extensions of Δ. Moreover, the set of preferred (resp., stable, semi-stable, grounded, ideal) extensions is defined in the standard way (see Section 2.1) by using again $\textsc{Def}(\mathbf{S})$ and $\textsc{Acc}(\mathbf{S})$.

Using $\textsc{Acc}(\mathbf{S})$, we define the propositional program for an RAFD Δ.

Definition 7. *Given an RAFD $\Delta = \langle A, \Sigma, \Pi, \mathbf{s}, \mathbf{t}\rangle$, then P_Δ (the propositional program derived from Δ) contains, for each $X \in A \cup \Sigma \cup \Pi$, a rule of the form*

$$X \leftarrow \bigwedge_{\alpha \in \Sigma \wedge \mathbf{t}(\alpha) = X} (\neg\alpha \vee \neg \mathbf{s}(\alpha)) \wedge \bigwedge_{\beta \in \Pi \wedge \mathbf{s}(\beta) = X} (\neg\beta \vee \mathbf{t}(\beta)).$$

Theorem 4. *For any RAFD Δ, $\widehat{\mathcal{CO}(\Delta)} = \mathcal{PS}(P_\Delta)$.*

Thus, the semantics of an RAFD Δ can be carried out by using the PSMs of P_Δ. The result also applies to restricted frameworks such as RAF, where $\Pi = \emptyset$, and AFD, where $\mathbf{t} : \Sigma \cup \Pi \to A$.

AFRAD. The following definition formalizes defeated and acceptable sets for an AFRAD.

Definition 8. *Given an AFRAD $\langle A, \Sigma, \Pi, \mathbf{s}, \mathbf{t}\rangle$ and a set $\mathbf{S} \subseteq A \cup \Sigma \cup \Pi$, we have*

- $\textsc{Def}(\mathbf{S}) = \{X \in A \cup \Sigma \cup \Pi \mid (X \in \Sigma \wedge \mathbf{s}(X) \in \textsc{Def}(\mathbf{S})) \vee (\exists \alpha \in \Sigma \cap \mathbf{S} \,.\, \mathbf{t}(\alpha) = X) \vee (\exists \beta \in \Pi \cap \mathbf{S} \,.\, \mathbf{s}(\beta) = X \wedge \mathbf{t}(\beta) \in \textsc{Def}(\mathbf{S}))\}$;
- $\textsc{Acc}(\mathbf{S}) = \{X \in A \cup \Sigma \cup \Pi \mid (X \in \Sigma \Rightarrow \mathbf{s}(X) \in \textsc{Acc}(\mathbf{S}))) \wedge (\forall \alpha \in \Sigma \,.\, \mathbf{t}(\alpha) = X \Rightarrow \alpha \in \textsc{Def}(\mathbf{S})) \wedge (\forall \beta \in \Pi \,.\, \mathbf{s}(\beta) = X \Rightarrow (\beta \in \textsc{Def}(\mathbf{S}) \vee \mathbf{t}(\beta) \in \textsc{Acc}(\mathbf{S}))))\}$.

Similarly to what done for RAFDs, the set $\mathcal{CO}(\Delta)$ of complete extensions of an AFRAD Δ, and the sets of extensions prescribed by the other semantics, are defined by using the defeated and acceptable sets defined above.

Definition 9. *For any AFRAD $\Delta = \langle A, \Sigma, \Pi, \mathbf{s}, \mathbf{t}\rangle$, P_Δ (the propositional program derived from Δ) contains, for each $X \in A \cup \Sigma \cup \Pi$, a rule of the form*

$$X \leftarrow \varphi(X) \wedge \bigwedge_{\alpha \in \Sigma \wedge \mathbf{t}(\alpha) = X} \neg\alpha \wedge \bigwedge_{\beta \in \Pi \wedge \mathbf{s}(\beta) = X} (\neg\beta \vee \mathbf{t}(\beta))$$

where: $\varphi(X) = \mathbf{s}(X)$ if $X \in \Sigma$; otherwise, $\varphi(X) = \mathtt{true}$.

Theorem 5. *For any AFRAD Δ, $\widehat{\mathcal{CO}(\Delta)} = \mathcal{PS}(P_\Delta)$.*

The result also apply to restricted frameworks such as AFRA and AFD.

Example 5. Consider now the Rec-BAF Δ' of Example 4 and assume that supports are interpreted as deductive. The propositional program under RAFD semantics is as follows: $P_{\Delta'} = \{(\mathtt{o} \leftarrow \neg\beta_1 \vee \alpha_1), (\mathtt{r} \leftarrow \neg\alpha_3 \vee \neg\mathtt{s}), (\mathtt{p} \leftarrow \neg\alpha_1 \vee \neg\mathtt{r}), (\mathtt{s} \leftarrow \neg\alpha_2 \vee \neg\mathtt{r}),$

$(\alpha_1 \leftarrow), (\alpha_2 \leftarrow), (\alpha_3 \leftarrow), (\beta_1 \leftarrow)\}$, whose set of partial stable models is $\mathcal{PS}(P_{\Delta'}) = \{M_1 = \{\mathtt{o}, \neg\mathtt{r}, \mathtt{p}, \mathtt{s}, \alpha_1, \alpha_2, \alpha_3, \beta_1\}, M_2 = \{\mathtt{o}, \mathtt{r}, \neg\mathtt{p}, \neg\mathtt{s}, \alpha_1, \alpha_2, \alpha_3, \beta_1\}\}$.

On the other hand, the propositional program for Δ' under AFRAD semantics is: $P_{\Delta'} = \{(\mathtt{o} \leftarrow \neg\beta_1 \vee \alpha_1), (\mathtt{r} \leftarrow \neg\alpha_3), (\mathtt{p} \leftarrow \neg\alpha_1), (\mathtt{s} \leftarrow \neg\alpha_2), (\alpha_1 \leftarrow \mathtt{r}), (\alpha_2 \leftarrow \mathtt{r}), (\alpha_3 \leftarrow \mathtt{s}), (\beta_1 \leftarrow)\}$, whose set of partial stable models is $\mathcal{PS}(P_{\Delta'}) = \{M_3 = \{\mathtt{o}, \neg\mathtt{r}, \mathtt{p}, \mathtt{s}, \neg\alpha_1, \neg\alpha_2, \alpha_3, \beta_1\}, M_4 = \{\mathtt{o}, \mathtt{r}, \neg\mathtt{p}, \neg\mathtt{s}, \alpha_1, \alpha_2, \neg\alpha_3, \beta_1\}\}$.

Note that the RAFD (resp., AFRAD) program differs from the RAFN (resp., ASAF) program given in Example 4 only in rules having as head α_1 and $\mathtt{o}$, respectively. □

Considering Example 4 and Example 5, we note that the set of PSMs of the propositional programs for the RAFN and RAFD (resp., ASAF and AFRAD) frameworks coincide, meaning that these argumentation frameworks have the same sets of extensions. However, in general, RAFNs and RAFDs (resp., ASAFs and AFRADs) prescribe different sets of extensions as shown in the following example.

Example 6. Consider a Rec-BAF Δ'' obtained by adding to the Rec-BAF Δ' of Example 5 a new argument $\mathtt{z}$ and a new support β_2 from argument $\mathtt{p}$ to $\mathtt{z}$ (see Figure 3).

We first revise the propositional programs given in Example 4 to take into account argument $\mathtt{z}$ and support β_2. The propositional program $P_{\Delta''}$ under RAFN (resp., ASAF) semantics is obtained by adding to the propositional program $P_{\Delta'}$ under RAFN (resp., ASAF) semantics of Example 4 the rules $\mathtt{z} \leftarrow \neg\beta_2 \vee \mathtt{p}$ and $\beta_2 \leftarrow$ (cf. Definitions 3 and 5). Then, the set of partial stable models of $P_{\Delta''}$ under RAFN semantics is: $\mathcal{PS}(P_{\Delta''}) = \{N_1 = M_1 \cup \{z, \beta_2\}, N_2 = M_2 \cup \{\neg z, \beta_2\}\}$, where M_1 and M_2 are the models given in Example 4 that, as said above, coincide with those of Example 5. The set of partial stable models of $P_{\Delta''}$ under ASAF semantics is as follows: $\mathcal{PS}(P_{\Delta''}) = \{N_3 = M_3 \cup \{z, \beta_2\}, N_4 = M_4 \cup \{\neg z, \beta_2\}\}$, where M_3 and M_4 are the models given in Example 4 which, again, coincide with those of Example 5.

Now consider how we have to revise the propositional programs of Example 5 to take into account the new elements $\mathtt{z}$ and β_2. In this case, the propositional program $P_{\Delta''}$ under RAFD (resp., AFRAD) semantics is obtained by adding to the propositional program $P_{\Delta'}$ under RAFD (resp., AFRAD) semantics of Example 5 the rules $\mathtt{z} \leftarrow$ and $\beta_2 \leftarrow$ and by replacing the rule having as head $\mathtt{p}$ with the rule $\mathtt{p} \leftarrow (\neg\alpha_1 \vee \neg\mathtt{r}) \wedge (\neg\beta_2 \vee \mathtt{z})$ (resp., $\mathtt{p} \leftarrow \neg\alpha_1 \wedge (\neg\beta_2 \vee \mathtt{z})$). Thus, set of partial stable models of $P_{\Delta''}$ under RAFD semantics is $\mathcal{PS}(P_{\Delta''}) = \{N_1 = M_1 \cup \{z, \beta_2\}, N_2' = M_2 \cup \{z, \beta_2\}\}$, while under AFRAD semantics we obtain $\mathcal{PS}(P_{\Delta''}) = \{N_3 = M_3 \cup \{z, \beta_2\}, N_4' = M_4 \cup \{z, \beta_2\}\}$.

Hence, here we have that PSM N_2 (resp., N_4) derived from RAFN (resp., ASAF) Δ'' differs from PSM N_2' (resp., N_4') derived from RAFD (resp., AFRAD) Δ''. Consequently, the sets of the extensions of the four frameworks differ as well. □

4 Conclutions and Future Work

By exploring the connection between formal argumentation and logic programming, we have proposed a simple but general logical framework which is able to capture, in a systematic and succinct way, the different features of several AF-based frameworks under different argumentation semantics and interpretation of the support relation. The proposed approach can be used for better understanding the semantics of extended AF

frameworks (sometimes a bit involved), and is flexible enough for encouraging the study of other extensions. Our approach can be used to provide additional tools for computing complete extensions using answer set solvers [25] and classical program rewriting [30, 35, 24]. For instance, ASP-based tools like DLV and Potassco can be used to compute stable models, and XSB can be used to compute well founded semantics.

Other extensions of the Dung's framework not explicitly discussed in this paper are also captured by our technique as they are special cases of some of those studied in this paper. This is the case of *Extended AF (EAF)* and *hierarchical EAF*, which extend AF by allowing second order and stratified attacks, respectively [31].

Future work will be devoted to further generalize our logical approach in order to deal also with AF-based framework considering probabilities [23], weights [11], and preferences [8, 31], and frameworks considering supports with multiple sources [17]. Finally, we plan to investigate incremental techniques tailored at using our approach to compute extensions of dynamic AF-based frameworks, where the sets of arguments and interactions change over the time [28, 3, 4, 2, 5].

References

1. Alcântara, J.F.L., Sá, S., Guadarrama, J.C.A.: On the equivalence between abstract dialectical frameworks and logic programs. TPLP **19**(5-6), 941–956 (2019)
2. Alfano, G., Cohen, A., Gottifredi, S., Greco, S., Parisi, F., Simari, G.R.: Dynamics in abstract argumentation frameworks with recursive attack and support relations. In: ECAI. pp. 577–584 (2020)
3. Alfano, G., Greco, S., Parisi, F.: Efficient computation of extensions for dynamic abstract argumentation frameworks: An incremental approach. In: IJCAI. pp. 49–55 (2017)
4. Alfano, G., Greco, S., Parisi, F.: A meta-argumentation approach for the efficient computation of stable and preferred extensions in dynamic bipolar argumentation frameworks. Intelligenza Artificiale **12**(2), 193–211 (2018)
5. Alfano, G., Greco, S., Parisi, F.: An efficient algorithm for skeptical preferred acceptance in dynamic argumentation frameworks. In: IJCAI. pp. 18–24 (2019)
6. Alfano, G., Greco, S., Parisi, F., Trubitsyna, I.: On the semantics of abstract argumentation frameworks: A logic programming approach. TPLP **20**(5), 703–718 (2020)
7. Alfano, G., Greco, S., Parisi, F., Trubitsyna, I.: On the semantics of abstract argumentation frameworks: A logic programming approach. CoRR **abs/2008.02550** (2020)
8. Amgoud, L., Vesic, S.: A new approach for preference-based argumentation frameworks. Ann. Math. Artif. Intell. **63**(2), 149–183 (2011)
9. Baroni, P., Cerutti, F., Giacomin, M., Guida, G.: AFRA: Argumentation Framework with Recursive Attacks. IJAR **52**(1), 19–37 (2011)
10. Bench-Capon, T., Dunne, P.E.: Argumentation in artificial intelligence. AI **171**, 619 – 641 (2007)
11. Bistarelli, S., Rossi, F., Santini, F.: A novel weighted defence and its relaxation in abstract argumentation. IJAR **92**, 66–86 (2018)
12. Bondarenko, A., Dung, P.M., Kowalski, R.A., Toni, F.: An abstract, argumentation-theoretic approach to default reasoning. AI **93**, 63–101 (1997)
13. Caminada, M., Sá, S., Alcântara, J.F.L., Dvorák, W.: On the equivalence between logic programming semantics and argumentation semantics. IJAR **58**, 87–111 (2015)
14. Caminada, M., Schulz, C.: On the equivalence between assumption-based argumentation and logic programming. JAIR **60**, 779–825 (2017)

15. Cayrol, C., Lagasquie-Schiex, M.: Bipolarity in argumentation graphs: Towards a better understanding. IJAR **54**(7), 876–899 (2013)
16. Cayrol, C., Fandinno, J., del Cerro, L.F., Lagasquie-Schiex, M.: Valid attacks in argumentation frameworks with recursive attacks. In: COMMONSENSE (2017)
17. Cayrol, C., Fandinno, J., del Cerro, L.F., Lagasquie-Schiex, M.: Structure-based semantics of argumentation frameworks with higher-order attacks and supports. In: COMMA. pp. 29–36 (2018)
18. Cohen, A., Gottifredi, S., Garcia, A.J., Simari, G.R.: A survey of different approaches to support in argumentation systems. The Know. Eng. Rev. **29**(5), 513–550 (2014)
19. Cohen, A., Gottifredi, S., Garcia, A.J., Simari, G.R.: An approach to abstract argumentation with recursive attack and support. J. Appl. Log. **13**(4), 509–533 (2015)
20. Craven, R., Toni, F.: Argument graphs and assumption-based argumentation. AI **233**, 1–59 (2016)
21. Dung, P.M.: Negations as hypotheses: An abductive foundation for logic programming. In: ICLP. pp. 3–17 (1991)
22. Dung, P.M.: On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. AI **77**, 321–358 (1995)
23. Fazzinga, B., Flesca, S., Parisi, F.: On the complexity of probabilistic abstract argumentation frameworks. TOCL **16**(3), 22:1–22:39 (2015)
24. Gaggl, S.A., Manthey, N., Ronca, A., Wallner, J.P., Woltran, S.: Improved answer-set programming encodings for abstract argumentation. TPLP **15**(4-5), 434–448 (2015)
25. Gebser, M., Leone, N., Maratea, M., Perri, S., Ricca, F., Schaub, T.: Evaluation techniques and systems for answer set programming: a survey. In: IJCAI. pp. 5450–5456 (2018)
26. Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In: ICSLP. pp. 1070–1080 (1988)
27. Gottifredi, S., Cohen, A., Garcia, A.J., Simari, G.R.: Characterizing acceptability semantics of argumentation frameworks with recursive attack and support relations. AI **262**, 336–368 (2018)
28. Greco, S., Parisi, F.: Incremental computation of deterministic extensions for dynamic argumentation frameworks. In: JELIA. pp. 288–304 (2016)
29. Greco, S., Saccà, D.: Complexity and expressive power of deterministic semantics for datalog¬. Inf. Comput. **153**(1), 81–98 (1999)
30. Janhunen, T., Niemelä, I., Seipel, D., Simons, P., You, J.H.: Unfolding partiality and disjunctions in stable model semantics. ACM Trans. Comput. Logic **7**(1) (2006)
31. Modgil, S.: Reasoning about preferences in argumentation frameworks. AI **173**(9-10), 901–934 (2009)
32. Nouioua, F., Risch, V.: Argumentation frameworks with necessities. In: SUM. pp. 163–176 (2011)
33. Saccà, D.: The expressive powers of stable models for bound and unbound DATALOG queries. J. Comput. Syst. Sci. **54**(3), 441–464 (1997)
34. Saccà, D., Zaniolo, C.: Stable models and non-determinism in logic programs with negation. In: PODS. pp. 205–217 (1990)
35. Sakama, C., Rienstra, T.: Representing argumentation frameworks in answer set programming. Fundam. Inform. **155**(3), 261–292 (2017)
36. Simari, G.R., Rahwan, I. (eds.): Argumentation in Artificial Intelligence (2009)
37. Villata, S., Boella, G., Gabbay, D.M., van der Torre, L.W.N.: Modelling defeasible and prioritized support in bipolar argumentation. AMAI **66**(1-4), 163–197 (2012)
38. Wu, Y., Caminada, M., Gabbay, D.M.: Complete extensions in argumentation coincide with 3-valued stable models in logic programming. Studia Logica **93**(2-3), 383–403 (2009)