Defining the Semantics of Abstract Argumentation Frameworks through Logic Programs and Partial Stable Models (Extended Abstract)*

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 Argumentation Framework (AF) include the class of Recursive Bipolar AFs (Rec-BAFs), i.e. AFs with recursive attacks and supports. We show that a Rec-BAF Δ can be translated into a logic program PΔ so that the extensions of Δ under different semantics coincide with subsets of the partial stable models of PΔ.

1 Introduction
Dung’s abstract Argumentation Framework (AF) [Dung, 1995] is a simple yet powerful formalism for modelling disputes between two or more agents which 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 AF which allows for modelling the support between arguments [Nouioua and Risch, 2011; Villata et al., 2012]. Different interpretations of the notion of support have been proposed [Cayrol and Lagasquie-Schiex, 2013; Cohen et al., 2014]. Deductive support [Villata et al., 2012] is captured in 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 [Nouioua and Risch, 2011; Baroni et al., 2011] is interpreted in a dual way [Cayrol and Lagasque-Schiex, 2013]: 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 [Nouioua and Risch, 2011]; an AFD (AF with Deductive supports) is a BAF where supports are interpreted as deductions [Villata et al., 2012]. 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 AF consider recursive AFs and recursive BAFs where attacks/supports can be recursively at-tacked/supported [Baroni et al., 2011; Cayrol et al., 2017; Gottifredi et al., 2018; Cayrol et al., 2018].

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) [Baroni et al., 2011] and of an Attack-Support Argumentation Framework (ASAF) [Cohen et al., 2015; Gottifredi et al., 2018] 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 AFRA is defeated even when its source argument is defeated. This is not the case for Recursive AF (RAF) [Cayrol et al., 2017] and Recursive AF with Necessities (RAFN) frameworks [Cayrol et al., 2018], 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 the supports and different ways of determining the status of attacks.

Example 1 Consider a situation where we want to decide whether to play tennis on the basis of some information. Assume we have the following arguments: ɣ (it is raining), p (play tennis),  ν (the tennis court is outside),  s (it is sunny), and the implications: (ɣ1) if it is raining, then we do not play tennis, and (β1) if the tennis court is outside, then ɣ1 should hold. Notice that arguments ɣ and s are mutually conflicting. The scenario can be modelled using the Rec-BAF Δ of Figure 1 where ɣ1, as well as ɣ2 and ɣ3, modelling the mutual conflicts between ɣ and ɣ2, are attacks (denoted by →) and β1 is a support (denoted by ⇒).

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 [Wu et al., 2009].

Footnotes
*This paper is based on the work “On the semantics of abstract argumentation frameworks: A logic programming approach”, that was presented at the 36th International Conference on Logic Programming [Alfano et al., 2020c], awarded as best technical paper.
and further studied in [Caminada et al., 2015]. Efficient mappings from AF to Answer Set Programming (i.e. LP with Stable Model semantics [Gelfond and Lifschitz, 1988]) have been investigated as well [Sakama and Rienstra, 2017; Gagli et al., 2015]. However, the relationships between LP and more general frameworks extending AF such as Rec-BAFs has not been adequately studied. Thus, in this paper, by generalizing the work in [Caminada et al., 2015], 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 framework belonging to the class of Rec-BAFs, our formulation of acceptable elements allows us to easily and systematically derive a propositional program whose PSMs corresponds to the extensions (under a given semantics) of the given framework.

2 Preliminaries

We first recall Dung’s framework and then introduce the general class of recursive bipolar argumentation frameworks. We also briefly recall logic programs and partial stable semantics.

2.1 Argumentation Framework

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.

Different semantics notions have been defined leading to the characterization of collectively acceptable sets of arguments, called extensions [Dung, 1995]. Given an AF \( \Delta = \langle A, \Omega \rangle \) and a set \( S \subseteq A \) of arguments, an argument \( a \in A \) is said to be i) defeated w.r.t. \( S \) iff \( \exists b \in S \) such that \( (b, a) \in \Omega \), and ii) acceptable w.r.t. \( S \) iff for every argument \( b \in A \) with \( (b, a) \in \Omega \), there is \( c \in S \) such that \( (c, b) \in \Omega \). The sets of defeated and acceptable arguments w.r.t. \( S \) are defined as follows (where \( \Delta \) is understood):

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

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

Given an AF \( \langle A, \Omega \rangle \), a set \( S \subseteq A \) is an extension called: complete iff it is conflict-free and \( S = Acc(S) \); preferred iff it is a \( \subseteq \)-maximal complete extension; stable iff it is a total preferred extension, i.e. a preferred extension such that \( S \cup Def(S) = A \); semi-stable iff it is a preferred extension such that \( S \cup Def(S) \subseteq \subseteq \)-maximal; grounded iff it is the \( \subseteq \)-smallest complete extension; ideal iff it is the \( \subseteq \)-biggest complete extension contained in every preferred extension.

We use \( CO(\Delta) \) (resp., \( PR(\Delta) \), \( ST(\Delta) \), \( SST(\Delta) \), \( gR(\Delta) \), \( ID(\Delta) \)) to denote the set of complete (resp., preferred, stable, semi-stable, grounded, ideal) extensions of a framework \( \Delta \).

2.2 Recursive Bipolar AFs

A Recursive Bipolar Argumentation Framework (Rec-BAF) is a tuple \( \langle A, \Sigma, \Pi, s, t, \rangle \), where \( A \) is a set of arguments, \( \Sigma \) is a set of attack names, \( \Pi \) is a set of support names, \( s \) (resp., \( t \)) 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 \( \Phi \) such that either \( \Phi \subseteq \Sigma \) or \( \Phi \subseteq \Pi \), we denote by \( i \) \( \Phi^* = \{ (s(\gamma), t(\gamma)) | \gamma \in \Phi \} \) the set of pairs connected by an attack/support edge, and ii) \( \Phi + \) the transitive closure of \( \Phi \). It is assumed that \( \Pi^* \) is acyclic.

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

Recursive AF with Necessities. The Recursive Argumentation Framework with Necessities (RAFN) has been proposed in [Cayrol et al., 2018]. The semantics combines the RA interpretation of attacks [Cayrol et al., 2017] with that of BAF under the necessity interpretation of supports (i.e. AFN) [Nouioua and Risch, 2011]. Here we consider a simplified version where supports have a single source and the support relation is acyclic. Formally, given an RAFN \( \langle A, \Sigma, \Pi, s, t, \rangle \), \( X \in (A \cup \Sigma \cup \Pi) \), \( a \in A \), and \( S \subseteq A \cup \Sigma \cup \Pi \), we say that argument \( a \) recursively attacks \( X \) given \( S \) (denoted as \( a att_S X \)) if either \( (a, X) \in (\Sigma \cup \Pi)^* \) or there exists \( b \in A \) such that \( (a, b) \in (\Sigma \cup \Pi)^* \) and \( (b, X) \in (\Pi \cup \Sigma)^* \).

For any RAFN \( \Delta \) and \( S \subseteq A \cup \Sigma \cup \Pi \), the defeated and acceptable sets (given \( S \)) are defined as follows:

- \( \text{Def}(S) = \{ X \in A \cup \Sigma \cup \Pi | \exists b \in A \wedge b att_S X \} \);
- \( \text{Acc}(S) = \{ X \in A \cup \Sigma \cup \Pi | \forall b \in A \wedge b att_S X \Rightarrow b \notin \text{Def}(S) \} \).

Attack-Support AF. The Attack-Support Argumentation Framework (ASAF) has been proposed in [Cohen et al., 2015; Gottifredi et al., 2018]. The semantics combines the AFRA interpretation of attacks [Baroni et al., 2011] 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.

Formally, given an ASAF \( \langle A, \Sigma, \Pi, s, t, \rangle \), \( X \in (A \cup \Sigma \cup \Pi) \), \( a \in \Sigma \), and \( S \subseteq A \cup \Sigma \cup \Pi \), we say that i) \( a \) (directly or indirectly) attacks \( X \) (denoted by \( a def_X \)) if either \( t(a) = X \) or \( \alpha(a) = s(X) \), and ii) \( a \) extends \( X \) given \( S \) (denoted as \( a def_S X \)) if either \( a \) def X or there exists \( b \in A \) such that \( t(a) = b \) and either \( (b, X) \in (\Pi \cup \Sigma)^* \) or \( (b, s(X)) \in (\Pi \cup \Sigma)^* \). For any ASAF \( \Delta \) and \( S \subseteq A \cup \Sigma \cup \Pi \), the defeated and acceptable sets (given \( S \)) are as follows:

- \( \text{Def}(S) = \{ X \in A \cup \Sigma \cup \Pi | \exists a \in \Sigma \wedge a def_S X \Rightarrow a \notin \text{Def}(S) \} \);
- \( \text{Acc}(S) = \{ X \in A \cup \Sigma \cup \Pi | \forall a \in \Sigma \wedge a def_S X \Rightarrow a \in \text{Def}(S) \} \).

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 \( S \subseteq A \cup \Sigma \cup \Pi \) and by using the definitions of defeated and acceptable sets reported above.

Example 2 For the framework \( \Delta \) of Example 1, under RAFN semantics \( CO(\Delta) = \{ E_1 = \{ \alpha, s, p, \alpha_1, \alpha_2, \alpha_3, \beta_1 \}, E_2 = \{ \alpha, r, \alpha_1, \alpha_2, \alpha_3, \beta_1 \} \} \). Differently, under ASAF semantics \( CO(\Delta) = \{ E_3 = \{ \alpha, s, p, \alpha_3, \beta_1 \}, E_4 = \{ \alpha, r, \alpha_1, \alpha_2, \beta_1 \} \} \). Note that \( \alpha_1 \) and \( \alpha_2 \) (resp., \( \alpha_3 \)) do not belong to \( E_3 \) (resp. \( E_4 \)) since their source, \( r \) (resp., \( s \)), is defeated.
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 [Cohen et al., 2015] that in turns can be translated into an AF [Nouioua and Risch, 2011; Gottifredi et al., 2018].

**AF-based frameworks belonging to the class of Rec-BAFs.** 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 frameworks sharing the BAF syntax; their semantics differ because they interpret supports in different ways. Recursive AF (Rec-BAF) is another syntactic class, where AFs are extended by allowing recursive attacks: AFRA and RAF are frameworks belonging to this class, which only differ in the determination of attacks’ status. Finally, ASAF and RAFN are two frameworks belonging to the general class Rec-BAF presented in this section, consisting in the extension of BAF with recursive attacks and supports.

Hereafter, we focus on necessary supports only, even if the approach discussed in next section has been extended to cope with deductive supports in two frameworks: Recursive Argumentation Framework with Deductive supports (RAFD), which extends RAF, and Argumentation Framework with Recursive Attacks and Deductive supports (AFRAD), which extends AFRA. AFDs are special cases of these frameworks. We refer the reader to [Alfano et al., 2020c] for details.

Thus, the class of Rec-BAF includes AFN, RAF, AFD, and RAF and their specializations AF, AFRA, RAF, AFN, and AFD. We use \( \tilde{\mathcal{A}} \) to denote the set of these 9 frameworks.

### 2.3 Propositional Logic Programs

Given a set of symbols \( \Lambda = \{a_1, ..., a_n\} \), a (propositional) program over \( \Lambda \) is a set of \( \Lambda \) rules \( a_i \leftarrow \text{body}_{a_i} \), \( 1 \leq i \leq n \), where every \( \text{body}_{a_i} \) is a propositional formula defined over \( \Lambda \). The semantics of a propositional program \( P \), defined over a given alphabet \( \Lambda \), is given in terms of the set \( \mathcal{PS}(P) \) of its Partial Stable Models (PSMs) [Saccà and Zaniolo, 1990].

The set of PSMs of a logic program \( P \), denoted by \( \mathcal{PM}(P) \), define a meet-semilattice. The well-founded model (denoted by \( \mathcal{W}F(P) \)) and the maximal-stable models \( \mathcal{MS}(P) \), 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 [Saccà, 1997; Greco and Saccà, 1999].

3 AF-based Frameworks and PSMs

We present a new way to define the semantics of AF-based frameworks by considering propositional programs and PSMs. To compare extensions \( \tilde{E} \) of a given framework \( \Delta \) (containing acceptable elements) with PSMs of a given program \( P \) (containing true and false atoms), we denote by \( \tilde{E} = E \cup \{\neg a \mid a \in \text{Def}(E)\} \) the completion of \( E \). For a collection of extensions \( \tilde{E} \), \( \tilde{E} \) denotes the set \( \{E \mid E \in \tilde{E}\} \).

A first relationship between AF-based frameworks (e.g. BAF, Rec-BAF) and logic programs with PSMs is as follows.

**Proposition 1** For any framework \( \Delta \in \tilde{\mathcal{A}} \) and propositional program \( P \), whenever \( \mathcal{CO}(\Delta) = \mathcal{PS}(P) \) it holds that \( \mathcal{PR}(\Delta) = \mathcal{MS}(P) \), \( \mathcal{ST}(\Delta) = \mathcal{ST}(P) \), \( \mathcal{SST}(\Delta) = \mathcal{LM}(P) \), \( \mathcal{GR}(\Delta) = \mathcal{WF}(P) \), and \( \mathcal{TD}(\Delta) = \mathcal{MD}(P) \).

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 \( \Delta \) and an extension \( E \), for any element \( a \) which could occur in some extension of \( \Delta \), the truth value \( v_E(a) \) or simply \( v(a) \) whenever \( E \) is understood, is equal to true if \( a \in E \), false if \( a \in \text{Def}(E) \), undecided (undecided) otherwise. Hereafter, we assume that false < undecided < true and ¬undecided = undecided.

The 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 [Wu et al., 2009; Caminada et al., 2015]. The relationship is based on the observation that the meaning of an attack \( a \rightarrow b \) is that the condition \( v(b) \leq v(a) \) must hold. On the other side, the satisfaction of a logical rule \( a \leftarrow b, ..., b_n \) implies that \( v(a) \geq \min\{v(b_1), ..., v(b_n), \text{true}\} \).

**Definition 1** Given AF \( \Delta = \langle A, \Lambda \rangle \), \( P_\Delta = \{a \leftarrow \neg A_{a \in \Lambda} \rightarrow b \mid a \in A \} \) is the program derived from \( \Delta \).

The semantics of an AF \( \Delta \) can be obtained by considering PSMs of the logic program \( P_\Delta \). In fact, \( \mathcal{CO}(\Delta) = \mathcal{PS}(P_\Delta) \).

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. 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 \( \mathcal{S} \), will be denoted by \( \text{Def}(\mathcal{S}) \) and \( \text{Acc}(\mathcal{S}) \), respectively. These definitions will be used to derive rules in \( P_\Delta \), the propositional program for \( \Delta \in \tilde{\mathcal{A}} \). For AFs we have that, for every set \( \mathcal{S} \subseteq A \), \( \text{Def}(\mathcal{S}) = \text{Def}(\mathcal{S}) \) and \( \text{Acc}(\mathcal{S}) = \text{Acc}(\mathcal{S}) \).

3.1 Recursive BAFs with Necessary Supports

We now study the relationship between PSMs and the semantics of Rec-BAFs. We first present results for RAFN and logic programs with PSMs as follows.
RAFN. We next provide the definitions of defeated and acceptable sets for an RAFN.

Definition 2 For any RAFN \(\langle A, \Sigma, \Pi, s, t\rangle\) and a set \(S \subseteq A \cup \Sigma \cup \Pi\), we have that:

- **DEF(S)** = \(\{X \in A \cup \Sigma \cup \Pi \mid (\exists a \in S \cap t(a) = X) \vee (\exists \beta \in \Pi \cap s(\beta) \in \text{DEF}(S))\}\); 

- **ACC(S)** = \(\{X \in A \cup \Sigma \cup \Pi \mid (\forall \alpha \in \Sigma \cap t(\alpha) = X) \rightarrow (\alpha \in \text{DEF}(S) \cup s(\alpha) \in \text{ACC}(S))\}\). 

Notice that **DEF(S)** and **ACC(S)** are defined recursively. If \(S\) is a complete extension we obtain the following result.

Theorem 1 Given an RAFN \(\Delta\) and an extension \(S \in \text{CO}(\Delta)\), then \(\text{Def}(S) = \text{DEF}(S)\) and \(\text{Acc}(S) = \text{ACC}(S)\).

Theorem 1 states that in order to define the semantics for an RAFN \(\Delta\) we can use acceptable sets \(S = \text{ACC}(S)\). 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 \(\Delta = (A, \Sigma, \Pi, s, t)\), then \(P_\Delta\) (the propositional program derived from \(\Delta\)) contains, for each \(X \in A \cup \Sigma \cup \Pi\), a rule:

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

Intuitively, starting from the definition of **ACC(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 \(\alpha \) toward \(X\) is **false** or originates from a source \(s(\alpha)\) which is **false**, and (ii) every support \(\beta\) toward \(X\) is **false** or originates from a source \(s(\beta)\) which is **true**. These conditions resemble the conditions \((\forall \alpha \in \Sigma \cap t(\alpha) = X \Rightarrow (\alpha \in \text{DEF}(S) \cup s(\alpha) \in \text{DEF}(S))\) and \((\forall \beta \in \Pi \cap s(\beta) = X \Rightarrow (\beta \in \text{DEF}(S) \cup s(\beta) \in \text{ACC}(S))\), respectively, of Definition 2 after interpreting elements in **DEF(S)** as **false** and elements in **ACC(S)** as **true**.

As stated below, the set of complete extensions of an RAFN \(\Delta\) coincides with the set of PSMs of \(P_\Delta\).

Theorem 2 For any RAFN \(\Delta\), \(\text{CO}(\Delta) = \mathcal{PS}(P_\Delta)\).

Thus, using Proposition 1, also the others argumentation semantics turns out to be characterized in terms of subsets of PSMs.

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 to obtain a semantically-equivalent propositional program for ASAFs.

Definition 4 Given an ASAF \(\langle A, \Sigma, \Pi, s, t\rangle\) and a set \(S \subseteq A \cup \Sigma \cup \Pi\), we define:

- **DEF(S)** = \(\{X \in A \cup \Sigma \cup \Pi \mid (X \in \Sigma \cap s(\alpha) \in \text{DEF}(S)) \vee (\exists \alpha \in \Sigma \cap t(\alpha) = X) \vee (\exists \beta \in \Pi \cap s(\beta) = X \wedge s(\beta) \in \text{DEF}(S))\}\); 

- **ACC(S)** = \(\{X \in A \cup \Sigma \cup \Pi \mid (\forall \alpha \in \Sigma \cap t(\alpha) = X) \rightarrow (\alpha \in \text{DEF}(S) \wedge (\forall \beta \in \Pi \cap s(\beta) = X \rightarrow (\beta \in \text{DEF}(S) \vee s(\beta) \in \text{ACC}(S)))\}\). 

Analogously to what has been done RAFNs, since it can be shown that for any complete extension \(S\) of an ASAF \(\Delta\) it is the case that \(\text{Acc}(S) = \text{ACC}(S)\), the program \(P_\Delta\) for \(\Delta\) can be derived by looking at the definition **ACC(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 \(P_\Delta\). Specifically, the last two conjuncts in the definition of **ACC(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 attacker then all of its sources must be **true**.

Definition 5 For any ASAF \(\Delta = (A, \Sigma, \Pi, s, t)\), \(P_\Delta\) (the propositional program derived from \(\Delta\)) contains, for each \(X \in A \cup \Sigma \cup \Pi\), a rule of the form:

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

where: \(\varphi(X) = s(\alpha)\) if \(X \in \Sigma\); otherwise, \(\varphi(X) = \text{true}\).

The set of complete extensions of an ASAF \(\Delta\) coincides with the set of PSMs of \(P_\Delta\), 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 \(\Delta\), \(\text{CO}(\Delta) = \mathcal{PS}(P_\Delta)\).

The above results also apply to restricted frameworks such as AFRA, where \(\Pi = \emptyset\), and AFN, where \(t : \Sigma \cup \Pi \rightarrow A\).

Example 3 Consider \(\Delta\) of Example 1, whose complete extensions under RAFN and ASAF semantics are given in Example 2. The program under the RAFN semantics is \(P_\Delta = \{\langle o \rangle, (r \leftarrow \neg \alpha_3 \vee \neg s), (p \leftarrow \neg \alpha_1 \vee \neg r), (s \leftarrow \neg \alpha_2 \vee \neg r), (\alpha_2 \leftarrow \neg \beta_1 \vee o), (\alpha_2 \leftarrow, (\alpha_3 \leftarrow, (\beta_1 \leftarrow)\), \}\), whose set of PSMs is \(\mathcal{PS}(P_\Delta) = \{\{1\} = (o, r, s, p, \alpha_1, \alpha_2, \alpha_3, \beta_1)\}\).

For ASAF semantics, \(P_\Delta = \{\langle o \rangle, (r \leftarrow \neg \alpha_3), (p \leftarrow \neg \alpha_1), (s \leftarrow \neg \alpha_2), (\alpha_1 \leftarrow r \wedge \neg \beta_1 \vee o), (\alpha_2 \leftarrow r), (\alpha_3 \leftarrow, (\beta_1 \leftarrow)\), \}\), whose set of PSMs is \(\mathcal{PS}(P_\Delta) = \{(m) = (o, r, s, p, \alpha_1, \alpha_2, \alpha_3, \beta_1)\}\). Note that \(M_1\) (resp., \(M_2\)) differs from \(M_3\) (resp., \(M_4\)) in the status of \(\alpha_1\) and \(\alpha_2\) (resp., \(\alpha_3\)).

4 Conclusions and Future Work

We have proposed a general framework that can be used 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 [Gebser et al., 2018], as well as other complete-based semantics using classical program rewriting [Jannhunen et al., 2006; Sakama and Rienstra, 2017; Gaggl et al., 2015].

Future work will be devoted to generalize our approach to also deal with AF-based framework with probabilities [Fazzinga et al., 2015; Alfano et al., 2020a], weights [Bistarelli et al., 2018], preferences [Ampug and Vesic, 2011; Modgil, 2009], and constraints [Alfano et al., 2021b]. 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 [Alfano et al., 2020b; Alfano et al., 2021a].
References


