On the Abstract Reasoning Framework

(Discussion Paper)

Gianvincenzo Alfano, Sergio Greco, Francesco Parisi and Irina Trubitsyna

Department of Informatics, Modeling, Electronics and System Engineering (DIMES), University of Calabria, Rende, Italy

Abstract

Managing controversial information is a challenging issue in AI. Formal argumentation is an important area of AI concerned with reasoning about supporting and opposing arguments involved in decision-making processes. In abstract argumentation frameworks, each argument can be associated with an *acceptance condition*, which can be either implicit (e.g. in Dung's framework) or explicit (e.g. in the Dialectical Framework, where propositional formulas are associated with arguments/statements). However, existing argumentation frameworks that allow explicit conditions may not always offer a concise and intuitive way to express general acceptance conditions, such as those that can be expressed using first-order logic formulas. In this paper, we discuss the recently introduced *Abstract Reasoning Framework* (ARF) [1], that is an argumentation framework where arguments' acceptance conditions allow for checking general properties also concerning sets of arguments/statements by exploiting aggregate functions.

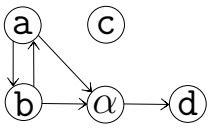
1. Introduction

Argumentation is used in our daily life to explain our reasons for or against claims in our discussions, persuade other people, and derive conclusions in a step-wise fashion. Most of the situations where argumentation takes place are inherently characterized by the presence of controversial information. Enabling automated systems to process such kind of information, much in the same way as organized human discussions are carried out, is an important challenge that have deserved increasing attention from the Artificial Intelligence community in the last decades. This has led to the development of an important and active research area called *formal argumentation* [2]. A central formalism in this area is Dung's abstract Argumentation Framework (AF) [3]. An AF consists of a set of *arguments* and a binary *attack* relation over the set of arguments that specifies the following relationship: if argument a attacks argument b, then b is acceptable only if a is not. Hence, arguments are abstract entities whose role is determined by attacks. We can think of an AF as a directed graph whose nodes represent arguments and edges represent attacks. The formal meaning of an AF is given in terms of argumentation semantics, e.g. the well-known *grounded*, *complete*, *preferred*, *stable*, and *semi-stable* semantics [3, 4], which intuitively tell us the sets of arguments (called *extensions*) that can collectively be accepted to support a point of view in a discussion, as illustrated in the following example.

Discussion Papers - 22nd International Conference of the Italian Association for Artificial Intelligence (AIxIA 2023), November 2023, Rome, Italy

g.alfano@dimes.unical.it (G. Alfano); greco@dimes.unical.it (S. Greco); fparisi@dimes.unical.it (F. Parisi); i.trubitsyna@dimes.unical.it (I. Trubitsyna)

Example 1. Suppose that *a party planner invites Antony, Bob, Carol, and David*. Due to their rivalry, Antony and Bob each replies that will join the party if the other does not. Carol replies that she will join the party anyway, and David replies that he will join the party if at least two among Antony, Bob, and Carol will be at the party. As the party planner is aware of the fact that Carol will join the party, they interpret David's reply as "he will join the party if Antony or Bob will do". This situation can be modeled by the AF shown in the the figure on the left-hand side, where each person's statement is represented by the name's initial and α is a meta-argument whose status is *false* (i.e. not accepted) if either Antony or Bob will join the party. According to the well-known preferred, stable, and semi-stable semantics [3, 4], we have that $\{\mathtt{a}, \mathtt{c}, \mathtt{d}\}$ and $\{\mathtt{b}, \mathtt{c}, \mathtt{d}\}$ are the preferred, stable, and semi-stable extensions of the AF shown in the figure. □



Despite the expressive power and generality of AFs, in some cases it is difficult to accurately model domain knowledge by an AF in a natural and easy-to-understand way. For this reason, Dung's framework has been extended by introducing further constructs, such as preferences [5, 6, 7, 8, 9], constraints [10, 11, 12, 13, 14], weights [15, 16], supports [17, 18], topics [19], and qualitative [20, 21, 22] and quantitative uncertainty [23, 24, 25, 26, 27] to achieve more comprehensive, natural, and compact ways for representing useful relationships among arguments [2]. However, many situations cannot be easily modeled using (extended) AFs. This happens, for instance, when the status of arguments may depend on a value obtained by aggregating some information. In fact, dealing with attacks and supports, it is not possible to state that an argument is accepted, and thus appears in an extension E under a given semantics, if the number of its supporters appearing in E is greater than the number of its attackers in E.

As an example, consider now the case where there are n people attending a party and that some pairs of people (e.g. male-female couples) attending together the party have the same ticket. Now person d says that they will attend the party if there are at least k pairs of people with the same ticket. To express such a condition in existing frameworks such as GRAPPA [28], several additional (meta-)statements have to be introduced to represent groups of 2 people sharing the same ticket among n people, thus possibly introducing $\binom{n}{2}$ new statements; in general, we would have $\binom{n}{m}$ new statements if groups of m people sharing the same ticket are considered. Moreover, considering the ADF framework [29] (a generalization of AF that allows explicit acceptance conditions over arguments using propositional formulae), it is not possible to naturally express acceptance conditions taking into account general properties on groups of objects that, for instance, can be easily expressed by using first-order logics. On the other side, the GRAPPA framework, that has firstly addressed the problem and proposed a solution, in some cases is not sufficiently flexible to easily express situations such as that mentioned above, as it could require the introduction of several additional statements, sometimes exponential.

Thus, in this paper we present the novel framework called *Abstract Reasoning Framework* (ARF) [1], allowing for versatile, easily understandable, and expressive acceptance conditions.

2. Preliminaries

We briefly summarize the basic concepts underlying the notion of partial stable models of logic programs [30], as ARF semantics relies on that.

A (normal, logic) program is a set of rules r of the form $a \leftarrow b_1 \wedge \cdots \wedge b_n$, with $n \geq 0$, where a is an atom, called head and denoted by $head(r)$, and $b_1 \wedge \cdots \wedge b_n$ is a conjunction of literals, called body and denoted by $body(r)$. With a little abuse of notation, $body(r)$ also denotes the set of literals in the body of r. 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 by 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., $neg(I)$, $undef(I)$) stands for $I \cap B_P$ (resp., $\neg I \cap B_P$, $B_P \setminus (pos(I) \cup neg(I))$). I is *consistent* if $pos(I) \cap \neg neg(I) = \emptyset$, otherwise I is *inconsistent*. A set $I \subseteq B_P \cup \neg B_P$ is a (*partial*) *interpretation* of P if I is consistent; I is *total* if $pos(I) \cup neg(I) = B_P$. For any partial interpretation I of a program P, the atoms in $pos(I)$ (resp., in $neg(I)$, $undef(I)$) are said to be *true* (resp., *false*, *undefined*) w.r.t. I. The truth value, either $\mathbf{t}$ (true), $\mathbf{f}$ (false) or $\mathbf{u}$ (undefined), of an atom a w.r.t. an interpretation I is denoted by $\vartheta_I(a)$. We assume the truth values ordering $\mathbf{f} < \mathbf{u} < \mathbf{t}$ and that $\neg\mathbf{u} = \mathbf{u}$.

A partial interpretation M of a program P is a *partial model* of P if for each rule $r \in ground(P)$, $\vartheta_M(head(r)) \geq \vartheta_M(body(r))$. Given a program P and a partial interpretation M, the *reduct* of P w.r.t. M, denoted by P^M, is obtained from $ground(P)$ by replacing each negated literal with its truth value in M. Clearly, rules having in the body the truth value $\mathbf{f}$ can be deleted, and the truth value $\mathbf{t}$ can be deleted from the body of rules.

As P^M is a positive program, the minimal Herbrand model of P^M can be obtained as the least fixpoint of its immediate consequence operator T_{P^M}, denoted by $T^{\omega}_{P^M}(\emptyset)$, containing true and undefined atoms. Let Ψ_{P^M} be the set of atoms which are either true or false w.r.t. $T^{\omega}_{P^M}(\emptyset)$ (false atoms are those in B_P that do not occur in $T^{\omega}_{P^M}(\emptyset)$ neither as true atoms nor as undefined atoms), then Ψ_{P^M} is minimal w.r.t. $pos(\Psi_{P^M})$ and maximal w.r.t. $neg(\Psi_{P^M})$.

Let P be a program and M a partial model of P. Then M is a *Partial Stable Model (PSM)* of P iff $M = \Psi_{P^M}$. The set of partial stable models of a logic program P define a meet semi-lattice. The *well-founded* model and the *maximal-stable* models are defined by considering the $\subseteq$-minimal and $\subseteq$-maximal elements. The set of (total) *stable* models is obtained by considering the maximal-stable models which are total, whereas the *least-undefined* (a.k.a. semi-stable) models are obtained by considering the maximal-stable models with a $\subseteq$-minimal set of undefined atoms.

The set of partial stable (resp., maximal-stable, (total) stable, least-undefined, well-founded) models of P will be denoted by $\mathcal{PS}(P)$ (resp., $\mathcal{MS}(P)$, $\mathcal{TS}(P)$, $\mathcal{LS}(P)$, $\mathcal{WS}(P)$).

Example 2. Consider the program $P = \{\mathtt{a} \leftarrow \neg\mathtt{b};\ \mathtt{b} \leftarrow \neg\mathtt{a};\ \mathtt{c} \leftarrow \neg\mathtt{a} \wedge \neg\mathtt{b} \wedge \neg\mathtt{d};\ \mathtt{d} \leftarrow \neg\mathtt{c}\}$. The set of PSMs of P is $\mathcal{PS}(P) = \{\emptyset, \{\neg\mathtt{c}, \mathtt{d}\}, \{\mathtt{a}, \neg\mathtt{b}, \neg\mathtt{c}, \mathtt{d}\}, \{\neg\mathtt{a}, \mathtt{b}, \neg\mathtt{c}, \mathtt{d}\}\}$. Then, $\mathcal{WS}(P) = \{\emptyset\}$, whereas $\mathcal{TS}(P) = \mathcal{MS}(P) = \mathcal{LS}(P) = \{\{\mathtt{a}, \neg\mathtt{b}, \neg\mathtt{c}, \mathtt{d}\}, \{\neg\mathtt{a}, \mathtt{b}, \neg\mathtt{c}, \mathtt{d}\}\}$. □

3. Abstract Reasoning Framework

We now present the syntax and the semantics of Abstract Reasoning Framework [1].

The language used to define acceptance conditions considers numerical terms, that is, natural numbers or terms built by using natural numbers, aggregate functions over set terms (defined

using general properties on groups of objects), and standard arithmetic functions. Moreover, in addition to user-defined atoms, built-in atoms constructed by using numerical terms and comparison predicates are also allowed. Specifically, the alphabet consists of the following sets:

- *Constants* $\mathcal{C}$, consisting of a set of user-defined objects (denoted by names starting with a lowercase letter) and the set $\mathbb{N}$ of natural numbers;
- *Variables* $\mathcal{V}$, denoted by names starting with an uppercase letter;
- *Functions* $\mathcal{F}$, consisting of arithmetic functions[1] $+, -, \times$ and aggregate functions min, max, $count$ and sum;
- *Predicates* Π, consisting of a set of user-defined predicates Q and the built-in predicates $>, \geq, <, \leq, =, \neq$ (also called comparison operators);
- *Logical connectives* $\mathcal{L} = \{\wedge, \neg\}$.

As illustrated below, we assume the existence of *set terms* that can be defined by specifying their properties (by a logical formula to be satisfied). Aggregate functions min, max, and sum are assumed to be applied to the first element of the tuples in the input set. Thus, in constructing formulae defining generalized acceptance conditions, we use atoms having as terms both simple terms and aggregate terms, that is aggregate functions applied to set terms.

The basic elements for building acceptance conditions are:

- *Simple terms*, that are constants and variables;
- *User-defined atoms*, built by using predicates in Q and simple terms as usual.
- *Built-in atoms* of the form $E_1 \odot E_2$, where E_1 and E_2 are arithmetic expressions built by using natural numbers and aggregate terms (see next) and $\odot$ is a comparison operator.
- *Literals*, consisting of built-in atoms or possibly negated user-defined atoms. Analogously to atoms, literals are distinguished between user-defined and built-in ones.
- *Set terms* of the form $\{\overline{X} \mid \varphi(\overline{X}, \overline{Z})\}$, where $\varphi(\overline{X}, \overline{Z})$ is a safe conjunction of literals[2] such that built-in atoms contain only simple terms; here $\overline{X}$ is a list of so-called *aggregate variables*, whereas $\overline{Z}$ is the list of remaining variables, called *existential variables*.
- *Aggregate terms* of the form $f(U)$, where $U = \{\overline{X} \mid \varphi(\overline{X}, \overline{Z})\}$ is a set term and f is an aggregate function.
- *Logical rules* having in the head a ground user-defined atom and in the body a conjunction of ground user-defined literals and built-in literals.

Notice that the above definitions do not allow to have nested aggregate terms (i.e. only simple terms can occur inside aggregate terms). W.l.o.g., we can assume that rules may have at most one built-in atom with aggregate terms. It is also assumed that: *i*) conjunctions inside set terms are *safe* [31], *ii*) variables occurring in an aggregate term do not occur elsewhere outside the aggregate term in the same rule, and *iii*) built-in atoms are *monotonic*. Intuitively, a built-in atom is monotonic if by enlarging the set terms occurring in it, its truth value does not decrease (e.g. if it was $\mathbf{t}$ it cannot become $\mathbf{f}$ after enlarging the set). These properties are discussed in detail in [1].

We are now ready to introduce the ARF syntax.

[1] The operator / has been excluded to avoid possible division by zero.

[2] Intuitively, safeness prescribes that variables take values from a finite domain.

Definition 1 (ARF Syntax). An Abstract Reasoning Framework (ARF) is a tuple $\langle O, Q, S, G \rangle$, where O is a finite set of objects, Q is a set of predicates, S is a finite set of statements built by using predicates in Q and constants in $\mathcal{C}$, and G is a finite set of logical rules defining all statements in S.

Example 3. Continuing with Example 1, assume now that Carol changes her mind and says that she will join the party if Bob does not. For the new party planning scenario, we obtain the following ARF $\Delta_3 = \langle \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}, \{\mathtt{j}\}, \{\mathtt{j(a)}, \mathtt{j(b)}, \mathtt{j(c)}, \mathtt{j(d)}\}, G_3 \rangle$, where $\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}$ respectively denote *Antony, Bob, Carol, David*, the 1-arity predicate $\mathtt{j}$ stands for *joins the party*, and G_3 is as follows:

$$\begin{array}{l|l} \mathtt{j(a)} \leftarrow \neg \mathtt{j(b)} & \mathtt{j(b)} \leftarrow \neg \mathtt{j(a)} \\ \mathtt{j(c)} \leftarrow \neg \mathtt{j(b)} & \mathtt{j(d)} \leftarrow \mathtt{count}\{\mathtt{X} \mid \mathtt{j(X)} \wedge \mathtt{X} \neq \mathtt{d}\} \geq 2. \end{array}$$

In the last rule, $\mathtt{j(X)}$ is a user-defined atom, whereas $\mathtt{X} \neq \mathtt{d}$ is a built-in atom containing only simple terms. On the other side, $\mathtt{count}\{\mathtt{X} \mid \mathtt{j(X)} \wedge \mathtt{X} \neq \mathtt{d}\} \geq 2$ is a (monotonic) built-in atom containing an aggregate term. Notably, for the more general situation described in the introduction, where $\mathtt{d}$ joins the party only if at least k people will do, in the rule defining $\mathtt{j(d)}$ we simply need to replace 2 with k. □

3.1. ARF Semantics

Considering the tight connection between ARF and logic programming, we propose a semantics based on an extension of that proposed in [32] for logic programs with aggregates under *total* stable model semantics. The semantics of an ARF $\Delta = \langle O, Q, S, G \rangle$, given by the partial stable models of G, is obtained in two steps: (i) first, a *dual set* of rules $G^{\diamond}$ derived from G by rewriting rules with aggregates is generated, and then (ii) for a given candidate partial stable model N, the *P-reduct* $G^{\diamond N}$ (a ground, positive, standard program where aggregates are replaced by conjunctions of atoms) is generated and it is checked that N is the fixpoint of the immediate consequence operator $T_{G^{\diamond N}}$.

Generating the dual set of rules $G^{\diamond}$

For any ARF $\Delta = \langle O, Q, S, G \rangle$, the *dual set* of rules $G^{\diamond}$ is derived from G by:

1. replacing every aggregate term $f\{\overline{X} | \varphi(\overline{X}, \overline{Z})\}$ with a fresh variable V (called *auxiliary variable*), and adding to the body of the rule where it appears an *auxiliary* atom $p(V)$ (where p is a fresh predicate);
2. adding for each auxiliary atom $p(V)$, used to replace an aggregate term $f\{\overline{X} \mid \varphi(\overline{X}, \overline{Z})\}$, the following two *aggregate rules*:

   $$r' : p(V) \leftarrow V = f\{\overline{X} \mid \varphi(\overline{X}, \overline{Z}) \equiv \mathbf{t}\},$$
   $$r'' : p(V) \leftarrow V = f\{\overline{X} \mid \varphi(\overline{X}, \overline{Z}) \not\equiv \mathbf{f}\},$$

   where the body atom $V = f\{\cdots\}$ is called *aggregate atom*.

Thus, rules' bodies of $G^\diamond$ consist of either a conjunction of literals not containing aggregate terms or a single aggregate atom. Intuitively, in the dual set $G^\diamond$, heads of rules containing an aggregate atom can be derived from instances of set terms whose truth value is *true* or *undefined*. For the logical connective $\equiv$ there are different 3-valued semantics; we refer here to the Lukasiewicz's logic stating that the truth value of $a \equiv b$ is *true* if the truth values of a and b coincide, *false* if one is *false* and the other one is *true*, *undefined* otherwise.

The difference between G and $G^\diamond$ is that in $G^\diamond$ we have introduced new auxiliary atoms to compute intermediate values related to aggregate terms.

Example 4. Considering the set of rules G_3 of Δ_3, the dual set of rules $G_3^\diamond$ is as follows:

$$\begin{array}{l|l} \mathtt{j(a)} \leftarrow \neg \mathtt{j(b)} & \mathtt{j'(V)} \leftarrow \mathtt{V} = \mathtt{count}\{\mathtt{X} \mid (\mathtt{j(X)} \wedge \mathtt{X} \neq \mathtt{d}) \equiv \mathbf{t}\} \\ \mathtt{j(b)} \leftarrow \neg \mathtt{j(a)} & \mathtt{j'(V)} \leftarrow \mathtt{V} = \mathtt{count}\{\mathtt{X} \mid (\mathtt{j(X)} \wedge \mathtt{X} \neq \mathtt{d}) \not\equiv \mathbf{f}\} \\ \mathtt{j(c)} \leftarrow \neg \mathtt{j(b)} & \\ \mathtt{j(d)} \leftarrow \mathtt{j'(V)} \wedge \mathtt{V} \geq 2 & \end{array}$$

where $\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}$ are constants, $\mathtt{V}$ is an auxiliary variable and $\mathtt{j'(V)}$ is an auxiliary atom. □

We now define the satisfiability of literals and aggregate atoms. To this end, for any ARF $\Delta = \langle O, Q, S, G \rangle$, we denote by $pground(G^\diamond)$ the program derived from $G^\diamond$ by replacing auxiliary variables with constants in $\mathbb{N}$ in all possible ways. Let H be the set of atoms occurring in the heads of rules in $pground(G^\diamond)$. A partial interpretation N for $G^\diamond$ is a consistent subset of $H \cup \neg H$, that is, for each auxiliary predicate p, N defines at most one atom $p(k)$ as true (contained in the interpretation) and at most one atom $p(k')$ as undefined.

For instance, considering Example 4, the partial interpretation $N = \{\mathtt{j(a)}, \neg\mathtt{j(b)}, \mathtt{j'(1)}\} \cup \{\neg\mathtt{j'(x)} \mid \mathtt{x} \in (\mathbb{N} \setminus \{\mathtt{1}\})\}$ is consistent, while $N' = \{\mathtt{j(a)}, \neg\mathtt{j(b)}, \mathtt{j'(1)}, \mathtt{j'(2)}\} \cup \{\neg\mathtt{j'(x)} \mid \mathtt{x} \in (\mathbb{N} \setminus \{\mathtt{1}, \mathtt{2}, \mathtt{3}, \mathtt{4}\})\}$ is not. Observe that interpretations for G are finite whereas interpretations for $G^\diamond$ can be infinite, though enumerable and finitely representable.

A partial interpretation M for an ARF $\langle O, Q, S, G \rangle$ is a consistent set such that $M \subseteq S \cup \neg S$.

Definition 2 (Satisfiability of literals and aggregate atoms). Let $\Delta = \langle O, Q, S, G \rangle$ be an ARF and let N be a partial interpretation for $G^\diamond$. Then:

- a user-defined, ground literal ℓ is true (resp., false, undefined) w.r.t. N if $\ell \in N$ (resp., $\neg\ell \in N$, neither $\ell \in N$ nor $\neg\ell \in N$);
- the truth value of a ground built-in literal is computed in the standard way and it is either true or false;
- an aggregate atom $k = f\{\overline{X} \mid \varphi(\overline{X}, \overline{Z})\}$, with $k \in \mathbb{N}$, is:
  - true w.r.t. N if, let k_t be the value of $f\{\overline{X} \mid \varphi(\overline{X}, \overline{Z}) \equiv \mathbf{t}\}$ computed over N, the arithmetic statement $k = k_t$ is true; for $f \in \{min, max\}$ it is also required that $|\{\overline{X} \mid \varphi(\overline{X}, \overline{Z}) \equiv \mathbf{t}\}| > 0$;
  - undefined w.r.t. N if, let k_u be the value of $f\{\overline{X} \mid \varphi(\overline{X}, \overline{Z}) \not\equiv \mathbf{f}\}$ computed over N, the arithmetic statement $k = k_t$ is false and the arithmetic statement $k = k_u$ is true; for $f \in \{min, max\}$ it is also required that $|\{\overline{X} \mid \varphi(\overline{X}, \overline{Z}) \not\equiv \mathbf{f}\}| > 0$;

– false w.r.t. N, otherwise.

Observe that if N is a total interpretation then literals and aggregate atoms are either *true* or *false* w.r.t. N. Indeed, any aggregate atom cannot be *undefined* because $\varphi(\overline{X}, \overline{Z}) \equiv \mathbf{t}$ is equivalent to $\varphi(\overline{X}, \overline{Z}) \not\equiv \mathbf{f}$.

Generating the P-reduct of $G^{\diamond}$

ARF's partial stable model semantics extends the total stable model semantics for normal programs with aggregates defined in [32] and is based on the novel concept of *P-reduct* of the dual program, defined as follows.

Definition 3 (P-Reduct). Given an ARF $\Delta = \langle O, Q, S, G \rangle$ and a partial interpretation N for $G^{\diamond}$, the P-reduct $G^{\diamond N}$ is obtained from $pground(G^{\diamond})$ by:

1. deleting:
   a) aggregate rules whose body atom $k = f(U)$ is false w.r.t. N,
   b) rules having in the body an auxiliary atom not defined by any rule (i.e. there are no rules having that atom in the head), and
2. replacing:
   a) every aggregate atom $k = f\{\overline{X} \mid \varphi(\overline{X}, \overline{Z}) \equiv \mathbf{t}\}$ that is true w.r.t. N with the conjunction $\bigwedge\{\varphi(\overline{x}, \overline{z}) \mid \varphi(\overline{x}, \overline{z})$ is true w.r.t. $N\}$;
   b) every aggregate atom $k = f\{\overline{X} \mid \varphi(\overline{X}, \overline{Z}) \not\equiv \mathbf{f}\}$ that is not false w.r.t. N with the conjunction $\bigwedge\{\varphi(\overline{x}, \overline{z}) \mid \varphi(\overline{x}, \overline{z})$ is not false w.r.t. $N\}$;
   c) every negative (user-defined) body literal and built-in literal with its truth value w.r.t. N;
   where $\overline{x}$ and $\overline{z}$ are lists of constants replacing the list of variables $\overline{X}$ and $\overline{Z}$, respectively.

Therefore, $G^{\diamond N}$ is derived from $pground(G^{\diamond})$ and N, after deleting useless rules (items 1.a and 1.b), by performing the same steps as done for standard programs (item 2.c), and by replacing aggregate rules with standard rules (items 2.a and 2.b). Clearly, rules in $G^{\diamond N}$ having in the body a truth value $\mathbf{f}$ may be omitted as they do not contribute to the computation of the minimum model. Therefore, for each auxiliary predicate p, there are at most two rules defining it in $G^{\diamond N}$.

It is worth noting that conjunctions $\varphi(\overline{x}_1, \overline{z}_1) \wedge \cdots \wedge \varphi(\overline{x}_k, \overline{z}_k)$ replacing aggregate atoms of the form $k = count\{\overline{X} \mid \varphi(\overline{X}, \overline{Y})\}$ contain exactly k elements $\varphi(\overline{x}_i, \overline{z}_i)$. Note also that if N is a total interpretation for $G^{\diamond}$, then items 2.a and 2.b of Definition 3 are equivalent. We are now ready to define models for an ARF.

Definition 4 (ARF Semantics). *An interpretation M for an ARF $\Delta = \langle O, Q, S, G \rangle$ is a partial stable model (PSM) for Δ if there exists an interpretation N for $G^{\diamond}$ such that: i) N is the minimal model of the P-reduct $G^{\diamond N}$, and ii) $M = N \cap (S \cup \neg S)$.*

Notice that for ARFs where G is a standard program (not containing aggregate terms), the definition of partial stable model coincides with that of logic programs, as $G^{\diamond}$ coincides with G and an interpretation for it has no auxiliary atoms.

Example 5. Considering the set of rules G_3 of Δ_3 (see Example 3), the dual set of rules $G_3^\diamond$ (see Example 4), and the the interpretation $N_1 = \{\mathtt{j(a)}, \neg\mathtt{j(b)}, \mathtt{j(c)}, \mathtt{j(d)}, \mathtt{j'(2)}\} \cup \{\neg\mathtt{j'(x)} \mid \mathtt{x} \in (\mathbb{N} \setminus \{2\})\}$ for $G_3^\diamond$, the P-reduct $G_3^{\diamond N_1}$ is:

$$\begin{array}{l|l} \mathtt{j(a)} \leftarrow \mathbf{t} & \mathtt{j(c)} \leftarrow \mathbf{t} \\ \mathtt{j(d)} \leftarrow \mathtt{j'(2)} \wedge 2 \geq 2 & \mathtt{j'(2)} \leftarrow \mathtt{j(a)} \wedge \mathtt{j(c)}. \end{array}$$

Observe that rule $\mathtt{j(b)} \leftarrow \mathbf{f}$ has been omitted as its body is false. Therefore, $M_1 = N_1 \cap (S \cup \neg S) = \{\mathtt{j(a)}, \neg\mathtt{j(b)}, \mathtt{j(c)}, \mathtt{j(d)}\}$ is a PSM for Δ_3, as N_1 is the minimal model of $G_3^{\diamond N_1}$. □

We use $\mathcal{PS}(\Delta)$ to denote the set of PSMs for Δ. Consistently with the terminology of Section 2, a *maximal-stable* model is a $\subseteq$-maximal PSM for Δ. A *(total) stable* model is a maximal-stable model for Δ that is total, and a *least-undefined* model is a maximal-stable model for Δ with a $\subseteq$-minimal set of undefined statements.

Example 6. ARF Δ_3 has 3 PSMs: $M_0 = \emptyset$, $M_1 = \{\mathtt{j(a)}, \neg\mathtt{j(b)}\ \mathtt{j(c)}, \mathtt{j(d)}\}$ and $M_2 = \{\neg\mathtt{j(a)}, \mathtt{j(b)}, \neg\mathtt{j(c)}, \neg\mathtt{j(d)}\}$. M_1 and M_2 are maximal-stable, stable, and least-undefined. However, it is not always true that the maximal-stable models are also (total) stable. Indeed, consider the ARF Δ_6 obtained from Δ_3 by replacing the rule $\mathtt{j(c)} \leftarrow \neg\mathtt{j(b)}$ with $\mathtt{j(c)} \leftarrow \neg\mathtt{j(b)} \wedge \neg\mathtt{j(c)}$. In this case we have that Δ_6 has 3 PSMs: $M_0 = \emptyset$, $M_1 = \{\mathtt{j(a)}, \neg\mathtt{j(b)}\}$ and $M_2 = \{\neg\mathtt{j(a)}, \mathtt{j(b)}, \neg\mathtt{j(c)}, \neg\mathtt{j(d)}\}$. M_1 is maximal-stable, but not (total) stable, while M_2 (total) stable and, therefore, also maximal-stable and least-undefined. □

4. Conclusions

We have discussed the Abstract Reasoning Framework (ARF), that is a simple yet expressive argumentation framework where arguments' acceptance conditions allow aggregates and whose semantics is built on basis of the proposal of [32] that naturally extends (total) stable model semantics for standard logic programs to logic programs with aggregates. Aggregates over set terms have been studied in different contexts such as databases, logic programming, and argumentation-based frameworks. Interestingly, it can be shown that ARF generalizes several existing AF-based frameworks (e.g. Bipolar AF [33], AFRA, ASAF, and Recursive AF [17, 34]), as their semantics could be expressed using partial stable models [35]. We point out that ARF without aggregate functions can be easily implemented by using classical ASP solvers [36, 37]. However, this cannot be carried out for the (full) ARF framework since current ASP solvers do not fully support general aggregates as proposed in [32]. Indeed, the big research effort on the extension of ASP languages with aggregate constructs has culminated with the proposal presented in [32]. We strongly believe that this semantics will be incorporated in the next generation ASP solvers so that the ARF framework could be easily implemented. Finally, in [1] it is shown that the complexity of credulous and skeptical reasoning in ARF (that is, checking whether a given statement holds in any/all models) is not higher than that of AF-based frameworks under several well-known non-deterministic semantics.

Acknowledgments. We acknowledge financial support from PNRR MUR project PE0000013-FAIR and Next Generation EU project ECS0000009 - Tech4You.

References

[1] G. Alfano, S. Greco, F. Parisi, I. Trubitsyna, On acceptance conditions in abstract argumentation frameworks, Inf. Sci. 625 (2023) 757–779.

[2] D. Gabbay, M. Giacomin, G. R. Simari, M. Thimm (Eds.), Handbook of Formal Argumentation, volume 2, College Public., 2021.

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

[4] P. Baroni, M. Caminada, M. Giacomin, An introduction to argumentation semantics, Knowledge Engineering Review 26 (2011) 365–410.

[5] L. Amgoud, C. Cayrol, On the acceptability of arguments in preference-based argumentation, in: Proc. of UAI, 1998, pp. 1–7.

[6] S. Modgil, H. Prakken, A general account of argumentation with preferences, Artificial Intelligence 195 (2013) 361–397.

[7] G. Alfano, S. Greco, F. Parisi, I. Trubitsyna, On preferences and priority rules in abstract argumentation, in: Proc. of IJCAI, 2022, pp. 2517–2524.

[8] G. Alfano, S. Greco, F. Parisi, I. Trubitsyna, Abstract argumentation framework with conditional preferences, in: Proc. of AAAI, 2023, pp. 6218–6227.

[9] G. Alfano, S. Greco, F. Parisi, I. Trubitsyna, Preferences and constraints in abstract argumentation, in: Proc. of IJCAI, 2023, pp. 3095–3103.

[10] S. Coste-Marquis, C. Devred, P. Marquis, Constrained argumentation frameworks, in: Proc. of KR, 2006, pp. 112–122.

[11] O. Arieli, Conflict-free and conflict-tolerant semantics for constrained argumentation frameworks, Journal of Applied Logic 13 (2015) 582–604.

[12] G. Alfano, S. Greco, F. Parisi, I. Trubitsyna, Argumentation frameworks with strong and weak constraints: Semantics and complexity, in: Proc. of AAAI, 2021, pp. 6175–6184.

[13] G. Alfano, S. Greco, F. Parisi, I. Trubitsyna, Epistemic abstract argumentation framework: Formal foundations, computation and complexity, in: Proc. of AAMAS, 2023, pp. 409–417.

[14] G. Alfano, S. Greco, F. Parisi, I. Trubitsyna, Stable semantics for epistemic abstract argumentation framework, in: Proc. of Arg&App@KR, 2023, pp. 11–25.

[15] S. Bistarelli, F. Santini, Well-foundedness in weighted argumentation frameworks, in: Proc. of JELIA, 2019, pp. 69–84.

[16] S. Bistarelli, F. Rossi, F. Santini, A novel weighted defence and its relaxation in abstract argumentation, International Journal of Approximate Reasoning 92 (2018) 66–86.

[17] C. Cayrol, A. Cohen, M. Lagasquie-Schiex, Higher-order interactions (bipolar or not) in abstract argumentation: A state of the art, IfCoLog Journal of Logics and their Applications 8 (2021) 1339–1436.

[18] M. G. E. Gonzalez, M. C. D. Budán, G. I. Simari, G. R. Simari, Labeled bipolar argumentation frameworks, Journal of Artificial Intelligence Research 70 (2021) 1557–1636.

[19] M. C. D. Budán, M. L. Cobo, D. C. Martínez, G. R. Simari, Proximity semantics for topic-based abstract argumentation, Information Sciences 508 (2020) 135–153.

[20] D. Baumeister, M. Järvisalo, D. Neugebauer, A. Niskanen, J. Rothe, Acceptance in incomplete argumentation frameworks, Artif. Intell. (2021) 103470.

[21] G. Alfano, S. Greco, F. Parisi, I. Trubitsyna, Incomplete argumentation frameworks:

Properties and complexity, in: Proc. of AAAI, 2022, pp. 5451–5460.
[22] G. Alfano, M. Calautti, S. Greco, F. Parisi, I. Trubitsyna, Explainable acceptance in probabilistic and incomplete abstract argumentation frameworks, Artif. Intell. 323 (2023) 103967.
[23] H. Li, N. Oren, T. J. Norman, Probabilistic argumentation frameworks, in: Proc. of TAFA, 2011, pp. 1–16.
[24] A. Hunter, Some foundations for probabilistic abstract argumentation, in: Proc. of COMMA, 2012, pp. 117–128.
[25] B. Fazzinga, S. Flesca, F. Parisi, On the complexity of probabilistic abstract argumentation frameworks, TOCL 16 (2015) 22:1–22:39.
[26] B. Fazzinga, S. Flesca, F. Parisi, On efficiently estimating the probability of extensions in abstract argumentation frameworks, Int. J. Approx. Reason. 69 (2016) 106–132.
[27] G. Alfano, M. Calautti, S. Greco, F. Parisi, I. Trubitsyna, Explainable acceptance in probabilistic abstract argumentation: Complexity and approximation, in: Proc. of KR, 2020, pp. 33–43.
[28] G. Brewka, S. Woltran, GRAPPA: A semantical framework for graph-based argument processing, in: Proc. of ECAI, 2014, pp. 153–158.
[29] G. Brewka, H. Strass, S. Ellmauthaler, J. P. Wallner, S. Woltran, Abstract dialectical frameworks revisited, in: Proc. of IJCAI, 2013, pp. 803–809.
[30] D. Saccà, The expressive powers of stable models for bound and unbound DATALOG queries, Journal of Computer and System Sciences 54 (1997) 441–464.
[31] S. Abiteboul, R. Hull, V. Vianu, Foundations of Databases, Addison-Wesley, 1995.
[32] M. Gelfond, Y. Zhang, Vicious circle principle, aggregates, and formation of sets in ASP based languages, Artificial Intelligence 275 (2019) 28–77.
[33] F. Nouioua, AFs with necessities: Further semantics and labelling characterization, in: Proc. of SUM, 2013, pp. 120–133.
[34] G. Alfano, S. Greco, F. Parisi, A meta-argumentation approach for the efficient computation of stable and preferred extensions in dynamic bipolar argumentation frameworks, Intelligenza Artificiale 12 (2018) 193–211.
[35] G. Alfano, S. Greco, F. Parisi, I. Trubitsyna, On the semantics of abstract argumentation frameworks: A logic programming approach, Theory and Practice of Logic Programming 20 (2020) 703–718.
[36] M. Gebser, R. Kaminski, B. Kaufmann, T. Schaub, Multi-shot ASP solving with clingo, Theory and Practice of Logic Programming 19 (2019) 27–82.
[37] W. T. Adrian, M. Alviano, F. Calimeri, B. Cuteri, C. Dodaro, W. Faber, D. Fuscà, N. Leone, M. Manna, S. Perri, F. Ricca, P. Veltri, J. Zangari, The ASP system DLV: advancements and applications, Künstliche Intelligenz 32 (2018) 177–179.