Bipolar Argumentation Frameworks with Explicit Conclusions: Connecting Argumentation and Logic Programming

Victor Hugo Nascimento Rocha[1,*,†], Fabio Gagliardi Cozman[1,*,†]

[1]*Escola Politécnica, Universidade de São Paulo, Av. Prof. Luciano Gualberto, 380 - Butantã, São Paulo - SP, 05508-010, Brazil*

Abstract

We introduce a formalism for bipolar argumentation frameworks that combines different proposals from the literature and results in a one-to-one correspondence with logic programming. We derive the correspondence by presenting translation algorithms from one formalism to the other and by evaluating the semantic equivalences between them. We also show that the bipolar model encapsulates distinct interpretations of the support relations studied the literature.

Keywords

Argumentation, Logic Programming, Bipolar Argumentation

1. Introduction

The ability to argue is essential to humans, as discussed in philosophy since ancient times, in contexts ranging from politics and law to science and arts [1, 2]. Within artificial intelligence, argumentation has been boosted by the seminal work of Dung (1995) on Abstract Argumentation Frameworks (AAFs), where each argument is understood as an abstract entity whose acceptance depends only on its attack relations to other arguments.

Since Dung's paper, the connections between argumentation frameworks and other non-monotonic reasoning formalisms has been investigated at length. One such connection, put forward by Dung himself, is to logic programming. That research agenda was further developed by Caminada et al. (2015), who managed to prove equivalences between several of the semantics used by both formalisms and to present translation algorithms between them — not all correspondences were obtained by them, however; in particular, the connection between logic programming and AAFs breaks down in the context of the latter semi-stable semantics. Proposals for enlarged AAFs have been made and their equivalence to logic programming has been explored. Particularly interesting here are the Claim-augmented Argumentation Frameworks [5, 6]. Such frameworks have each argument explicitly associated with a claim. This apparently minor change to AAFs leads to a nice translation to logic programming and allows for further semantic equivalences.

In a different direction, Dung's abstract argumentation frameworks have been extended with support relations [7]. And the similarities between logic programming and Bipolar Argumentation Frameworks, where supports interact with attacks, have also been noted, for instance by Alfano et al. (2020). Those authors have proposed algorithms that translate different kinds of argumentation frameworks, including bipolar ones, to logic programs in order to evaluate their semantic differences. In addition to the steps previously proposed by Caminada et al. (2015), Alfano et al. (2020) interpret the support relation through non-negative atoms of the logic program.

Other relevant proposals have studied connections between logic programming and various argumentation formalisms, for instance assumption-based ones [9, 10]. However, to the best of our knowledge, none of these previous proposals reaches a one-to-one correspondence between some family of argumentation frameworks expressive enough to convey bipolarity and some well-known logic programming formalism.

In this work we will address the relationship between logic programming and argumentation frameworks by combining existing proposals, in particular the ones by Dvorák and Woltran (2019) and by Alfano et al. (2020). In doing so, we reach the Bipolar Conclusion-augmented Argumentation Framework and prove (for well-formed and non-redundant frameworks) its one-to-one equivalence to normal logic programming. The translation algorithms between both formalisms are also introduced. The proposed framework is able to encapsulate different versions of the support relation in the literature.

In short, we show that a large family of bipolar argu-

NMR 2022: 20th International Workshop on Non-Monotonic Reasoning, August 07–09, 2022, Haifa, Israel

*Corresponding author.

†These authors contributed equally.

victor.hugo.rocha@usp.br (V. H. N. Rocha); fgcozman@usp.br (F. G. Cozman)
http://www.poli.usp.br/p/fabio.cozman (F. G. Cozman)
0000-0002-8984-7982 (V. H. N. Rocha); 0000-0003-4077-4935 (F. G. Cozman)

mentation frameworks *is* normal logic programming, and vice-versa.

Section 2 briefly goes over needed background: abstract argumentation frameworks and their bipolar extension; logic programs and their semantics. Section 3 starts with relevant results from the literature on the relationship between Dung's framework and logic programming and then introduces the conclusion-augmented argumentation frameworks, showing that they improve on previous results by refining the equivalence between various formalisms. The bipolar conclusion-augmented argumentation model is later used to obtain a one-to-one equivalence with logic programming while also offering different interpretations of support. A novel discussion about the framework and its relationship with other proposals in the literature is developed in Section 4; finally, Section 5 concludes and describes possible future work.

2. Argumentation Frameworks and Logic Programming

In this section we review argumentation frameworks, bipolar frameworks, logic programming, and some of their semantics.

2.1. Abstract Argumentation Frameworks (AAFs)

Dung's argumentation frameworks are based on arguments and attacks between them. Arguments are understood as abstract entities whose internal structure is not relevant.

Definition 1. *An AAF is a tuple $(\mathcal{A}, \mathcal{R})$, where $\mathcal{A}$ is the set of arguments and $\mathcal{R}$ is an attack relationship on $\mathcal{A} \times \mathcal{A}$.*

An attack from an argument A to another argument B, represented by $A \rightarrow B$, intuitively means that if A is accepted, B cannot be. An AAF can be represented as a graph structure, where nodes stand for arguments and edges as the attack relation between them (see example in Figure 1).

Dung defined semantics through extensions. The latter represent sets of arguments that are acceptable according to some criterion. In this text, however, we will use labelings to define semantics [11].

Definition 2. *A labeling $\mathcal{L}$ of an AFF is a function $\mathcal{L}$: $\mathcal{A} \rightarrow \{\mathsf{In}, \mathsf{Out}, \mathsf{Undecided}\}$.*

Some concepts needed later are:

Definition 3. *An argument $A \in \mathcal{A}$ is acceptable iff all arguments B such that $B \rightarrow A$ are not acceptable.*

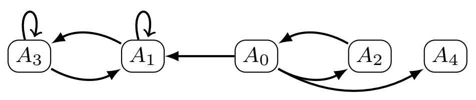

Figure 1: An Argumentation Graph as proposed by Dung.

Definition 4. *A labeling $\mathcal{L}$ of the arguments in an AFF is conflict-free iff there are no arguments A and B in the set of arguments labeled* In *for which $A \rightarrow B$.*

Definition 5. *A conflict-free labeling $\mathcal{L}$ of $\mathcal{A}$ is admissible iff for every argument A labeled* Out*, there exists an argument B labeled* In *such that $B \rightarrow A$.*

We can define several semantics using these concepts:

Definition 6. *An admissible labeling $\mathcal{L}$ of an AAF is complete iff for every argument A with the label* Undecided*, there are no arguments B with the label* In *that attack A and every acceptable argument with respect to the set of arguments* In *is also labeled* In*.*

Definition 7. *A complete labeling $\mathcal{L}$ of an AAF is preferred iff the argument set* In *is maximal.*

Definition 8. *A complete labeling $\mathcal{L}$ of an AAF is stable iff the argument set labeled* Undecided *is empty.*

Definition 9. *A complete labeling $\mathcal{L}$ of an AAF is grounded iff the argument set* In *is minimal.*

Definition 10. *A complete labeling $\mathcal{L}$ of an AAF is semi-stable iff the argument set* Undecided *is minimal.*

There are still other possible semantics [11], but in this paper we restrict ourselves to the previous ones.

2.2. Bipolar Argumentation Frameworks (BAFs)

An argumentation scenario seems to require not only attacks but also "positive" support relations between arguments [12]. The definition of support relations, unlike the attack relation, varies quite a bit in the literature [13, 14]. In this text we stick to the three interpretations explained by Cayrol and Lagasquie-Schiex (2013): the *necessary* support [15, 16], the *deductive* support [17], and the *evidential* support [18, 19]. A necessary support from one argument A to another B, represented by $A \Rightarrow_n B$, means that if B is accepted (received the label In), A must also be accepted. A deductive support from one argument A to another B, on the other hand, is represented by $A \Rightarrow_d B$, and means that if A is accepted (received the label In), B must also be accepted. Finally, there are two types of arguments in a BAF that contains evidential support: prima-facie arguments, which do not need any

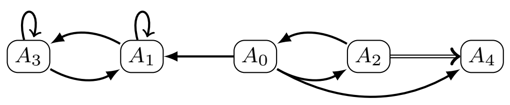

Figure 2: A Bipolar Argumentation Graph.

support to be accepted, and common arguments, which need to be supported by an accepted argument of the first type to be accepted.

Regardless of the interpretation of the support relation, a BAF can be defined as [7]:

Definition 11. *A Bipolar Argumentation Framework (BAF) is a tuple $(\mathcal{A}, \mathcal{R}_-, \mathcal{R}_+)$, where $\mathcal{A}$ represents the set of arguments, $\mathcal{R}_-$ the attack relation and $\mathcal{R}_+$ the support relation.*

As AFFs, BAFs can also be represented by graphs. Figure 2 depicts a BAF where nodes are arguments, edges encode the attack relation and double edges encode the support relation (for instance, A_2 supports A_4).

To handle support relations, the semantics, the labeling types and the acceptability criteria must be adapted. Several proposals have been made in order to achieve this, with differences in the way the relative strength between attack and support relations is taken into account [20, 21]. We will follow the proposal by Cayrol and Lagasquie-Schiex (2005).

We assume for now that support is of the necessary type; one can proceed analogously for the other types of support.

Once supports are taken into account, in addition to defeat by a traditional attack, an argument can be defeated indirectly.

Definition 12. *An argument B can be* defeated indirectly *by a sequence $A_1 R_1 \ldots R_{n-1} B$, where $i = 1, \ldots, n-2$, $R_i = R_+$ and $R_{n-1} = R_-$ or where $i = 2, \ldots, n-1$, $R_i = R_+$ and $R_1 = R_-$.*

Hence it makes sense to define the defeat/support of an argument by a set.

Definition 13. *Let $\mathcal{S} \subseteq \mathcal{A}$ and $A \in \mathcal{A}$. The set $\mathcal{S}$ defeats A iff there is a direct or indirect defeat for A from some element of $\mathcal{S}$. The set $\mathcal{S}$ supports A iff there is a sequence of the form $A_1 R_1 \ldots R_{n-1} A_n$, $n \geq 2$, such that $i = 1, \ldots, n-1$, $R_i = R_+$ with $A_n = A$ and $A_1 \in \mathcal{S}$.*

A set of arguments can also defend other arguments:

Definition 14. *Let $\mathcal{S} \subseteq \mathcal{A}$ and $A \in \mathcal{A}$. The set $\mathcal{S}$* collectively defends *A iff for some set $\mathcal{B} \subseteq \mathcal{A}$, if $\mathcal{B}$ defeats A then there is a $C \in \mathcal{S}$ such that C defeats $\mathcal{B}$.*

Given this, a conflict-free set and an admissible set can be redefined:

Definition 15. *Let $\mathcal{S} \subseteq \mathcal{A}$. The set $\mathcal{S}$ is conflict-free iff there are no $A, B \in \mathcal{S}$ such that A defeats (directly or indirectly) B.*

Definition 16. *Let $\mathcal{S} \subseteq \mathcal{A}$. The $\mathcal{S}$ set is admissible iff it is conflict-free and defends all its elements.*

We can then redefine the complete semantics for BAF:

Definition 17. *A set $\mathcal{S} \subseteq \mathcal{A}$ of arguments is complete iff it is admissible and every argument A that can be accepted together with $\mathcal{S}$ is part of $\mathcal{S}$.*

From the definition of complete semantics, the preferred, grounded, stable and semi-stable semantics can be adopted as before (as they are all special cases of the complete semantics in which some label is maximized or minimized).

2.3. Logic Programming and its Semantics

In this work we focus on propositional normal logic programs [4]:

Definition 18. *A (normal) logic program P is composed by a finite set of rules. Each rule r is an expression of the form $r : H :- A_1, \ldots, A_n, \textbf{not } B_1, \ldots, \textbf{not } B_n$, where H, A_i and B_i represent atoms and* **not** *is the classical negation. H represents the head of the formula, while the others are the body. A formula without the body is called a fact and is written as H. The Herbrand Base of the program P is the set HB_P of all atoms that appear in the program.*

Example 1. *The following is an example of a logic program with Herbrand Base consisting of the atoms a, b, c, d:*

$$r_0 : a :- \textbf{not } c, \qquad r_1 : b :- \textbf{not } a, \textbf{not } b,$$
$$r_2 : c :- \textbf{not } a, \qquad r_3 : b :- \textbf{not } b,$$
$$r_4 : d :- c, \textbf{not } a.$$

Definition 19. *A three-valued interpretation of a logic program $\mathbf{P}$ is a pair $I = (\mathcal{T}, \mathcal{F})$ such that $\mathcal{T} \cap \mathcal{F} = \emptyset$ and that both $\mathcal{T}$ and $\mathcal{F}$ contain elements from the Herbrand base of $\mathbf{P}$. $\mathcal{T}$ is understood as true, $\mathcal{F}$ as false and $HB_P \backslash (\mathcal{T} \cup \mathcal{F})$ as undecided.*

A three-valued model of $\mathbf{P}$ is an interpretation such that for each $a \in HB_P$ we have:

- *a is in $\mathcal{T}$ if there is a rule whose head is $a = H$ and where each A_i is in $\mathcal{T}$;*
- *a is in $\mathcal{F}$ if every rule whose head is $a = H$ is such that there is some A_i in $\mathcal{F}$.*

The reduct of $\mathbf{P}$ with respect to a three-valued interpretation $\mathcal{I}$, denoted $\mathbf{P}/\mathcal{I}$, is a logic program constructed using the following steps:

- *First, remove from $\mathbf{P}$ every rule that contains* $\mathbf{not}\ B$ *in its body for some* $B \in \mathcal{T}$*;*
- *Then, for each remaining rule, remove* $\mathbf{not}\ B$ *from the rule body if* $B \in \mathcal{F}$*;*
- *Finally, replace any remaining occurrences of* $\mathbf{not}\ B'$ *with a new* u *symbol representing "undecided".*

So $\mathbf{P}/\mathcal{I}$ *has a unique three-valued model with* $\mathcal{T}$ *minimum and* $\mathcal{F}$ *maximum (with respect to the set inclusion). We denote this model* $\Phi_{\mathbf{P}}(\mathcal{I})$*.*

It is then possible to define the semantics of a logic program P, given an interpretation $I = (\mathcal{T}, \mathcal{F})$, in several ways [4]:

Definition 20. *A partial stable (P-stable) model of* P *is an interpretation* I *such that* $\Phi_{\mathbf{P}}(\mathcal{I}) = I$*.*

Definition 21. *A model of* P *is well-founded iff it is P-stable and* $\mathcal{T}$ *is minimal.*

Definition 22. *A model of* P *is regular iff it is P-stable and* $\mathcal{T}$ *is maximal.*

Definition 23. *A model of* P *is stable iff it is P-stable and* $\mathcal{T} \cup \mathcal{F} = HB_P$*.*

Definition 24. *A model of* P *is L-stable iff it is P-stable and* $\mathcal{T} \cup \mathcal{F}$ *is maximal.*

The definitions above were crafted by Caminada et al. (2015) to emphasize connections with the corresponding argumentation semantics.

3. Correspondences between Argumentation Frameworks and Logic Programs

In this section we look at equivalences between argumentation frameworks and logic programming. We start by presenting results in the literature and then introduce the Conclusion-augmented Argumentation Frameworks (CAF), where a small change in the representation significantly improves the relationship between both formalisms. Furthermore, we expand CAFs by adding various interpretations of the support relation between arguments and discuss how those relations translate to logic programming.

3.1. The Connection between Abstract Argumentation Frameworks (AAFs) and Logic Programs

To study possible equivalences between AAFs and logic programs, one must consider translations from each other. A suitable starting point is the WCG (Wu, Caminada and Gabbay) algorithm [4], which is summarized below:

- Starting with a set of rules, process one rule at a time;
- If a rule of the form $H :- \mathbf{not}\ B_1, \ldots, \mathbf{not}\ B_n$ is found, then generate an argument A with *rules* $\{H :- \mathbf{not}\ B_1, \ldots, \mathbf{not}\ B_n\}$, *vulnerabilities* Vul($A$) $\{B_1, \ldots, B_n\}$, *conclusion* H and a set of sub-arguments that contain only A itself ;
- If a rule of the form $H :- A_1, \ldots, A_m, \mathbf{not}\ B_1, \ldots, \mathbf{not}\ B_n$ is found and assuming that each A_i has an associated argument Arg_i, then generate an argument A with a set of sub-arguments Arg_i, *conclusion* H, *rules* composed by the union of $\{H :- A_1, \ldots, A_m, \mathbf{not}\ B_1, \ldots, \mathbf{not}\ B_n\}$ with the rules of each sub-argument and *vulnerabilities* Vul(A) as the union of $\{B_1, \ldots, B_n\}$ with the vulnerabilities of the sub-arguments;
- After going through all the rules, the relations between arguments are established. If an argument A has a conclusion that is present in the vulnerabilities of another argument B, then A attacks B. With this, the AAF is created.

Example 2. *If we apply the WCG algorithm described above to the rules in Example 1, we obtain the AAF shown in Figure 1.*

Once the logic program is translated into an argumentation framework, we can apply any semantics to the latter and obtain a labeling $\mathcal{L}$ at the argument level. To then obtain the atom level (or "conclusion" level) labeling of P, the following mapping [4] can be used: the labeling of a conclusion is the one with the highest value among the arguments that are associated with it. The order of values between the labels is given by $\mathsf{In} > \mathsf{Undecided} > \mathsf{Out}$ and the idea behind it is that each conclusion is represented by the argument that best defends it.

We then ask: if we apply some semantics at the argument level and map the labeling to the conclusions, is the result equivalent to applying some other semantics directly to the logic program? The answer to this question is positive as was shown in [4].

Theorem 1. *(Theorems 19, 20, 21 and 22 of [4]). The labels of conclusions obtained by the semantics P-stable, regular, well-founded and stable in a logic program are equivalent to those obtained by the complete, preferred, grounded and stable semantics in an AFF respectively, with the subsequent transformation to the conclusions level.*

The attentive reader may have noticed that there is a notable exception in Theorem 1. There is no equivalence between the semi-stable and L-stable semantics. Intuitively, this happens because all other semantics are special cases of the P-stable models in logic programs and of the complete semantics in argumentation, with one of the labels maximized or minimized. For the labels In and

Out this last process is equivalent both at the argument and at the conclusion levels, but the same is not true for the label Undecided. Consider the following example:

Example 3. *If we apply the complete semantics to the AAF in Figure 1, we get the following three solutions (expressed as (*In *set,* Undecided *set,* Out *set)): (∅, {A_0, A_1, A_2, A_3, A_4}, ∅), ({A_0}, {A_3}, {A_1, A_2,A_4}) and ({A_2, A_4}, {A_1, A_3}, {A_0}). From this, we obtain the conclusion labelings: (∅, {a, b, c, d}, ∅), ({a}, {b}, {c, d}) and ({c, d}, {b}, {a}). As it can be seem, if the label* In *is maximized/minimized at both argument and conclusion levels, we obtain the same result. The same applies for the label* Out*. However, for the label* Undecided*, at the argument level we obtain only one solution by minimizing the label, while at the conclusion level we get two solutions.*

Despite this unfortunate feature of the semi-stable semantics, it is our position that the semi-stable semantics is the most appropriate semantics for argumentation. Compare with the other semantics. The grounded semantics, in situations of mutual attacks, does not reach any decisions, while the preferred semantics is very permissive. And the stable semantics may fail to produce a labeling when an Undecided label is unavoidable — in an argumentative scenario however, we believe that not arriving at some labeling is undesirable. We would thus like to choose the semi-stable semantics, but this seems to clash with our desire to obtain a correspondence between argumentation frameworks and logic programs. In the next subsection we show how to enlarge argumentation frameworks to obtain our desired correspondences.

3.2. The Connection between Conclusion-Augmented Argumentation Frameworks (CAFs) and Logic Programs

The first step in our pursuit of a complete correspondence between argumentation frameworks and logic programs is to augment arguments with their associated conclusions. We do so to guarantee that semi-stable semantics does have a correspondence in logic programming.

To do so, we adopt recent work on *Claim-augmented Argumentation Frameworks (CAFs)*, where each argument is augmented with its associated claim [5, 22, 23, 6]. We prefer to use "conclusion" rather than "claim" as the former term is employed in most of the previous literature dealing with connections between argumentation frameworks and logic programs (for instance, by Caminada et al. (2015)). So we will use *Conclusion-augmented Argumentation Frameworks (CAFs)*, but we emphasize that, despite the slight nomenclature change, the latter are equivalent to Claim-augmented Argumentation Frameworks. However we note that the previous work on CAFs

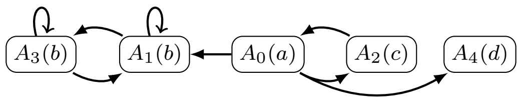

Figure 3: A Conclusion-Augmented Argumentation Graph

seems to take a more complicated route than ours to arrive at them and to study them; we thus describe our own route in some detail in this section, even though we acknowledge that the results are equivalent to previous ones by Dvorák and Woltran, Dvorák et al., Dvořák et al., Rapberger (2019, 2020, 2020, 2020). We also note that CAFs have been employed in recent work on probabilistic argumentation frameworks [24].

CAFs are defined as follows:

Definition 25. *A Conclusion-augmented Argumentation Framework (CAF) is a tuple* $(\mathcal{A}, \mathcal{S}, f, \mathcal{R})$*, with* $\mathcal{A}$ *being a finite set of arguments,* $\mathcal{S}$ *a set of conclusions,* f *is a function of arguments to conclusions, and* $\mathcal{R} \subseteq \mathcal{A} \times \mathcal{A}$ *is the attack relation as previously defined.*

We then say that every conclusion $c \in \mathcal{S}$ is associated to a finite set of arguments $f^{-1}(c)$. Analogously to AAFs, a conclusion-augmented argumentation graph is a directional graph in which each node represents an argument associated with its conclusion and each arrow represents an attack. It is important to note that the CAF maintains a high degree of abstraction as the internal structure of the arguments is not explicit. The only change is that each argument A is represented along with its conclusion a.

To each conclusion $c \in \mathcal{S}$ we can assign the labels In, Undecided, and Out. Naturally, this labeling of conclusions is related to the labeling of arguments through $f(A)$, and the procedure described previously obtains a labeling of conclusions from one of arguments.

A CAF can be generated by a slightly altered version of the WGU algorithm presented in Section 3.1, which associates each argument with a conclusion at the end of the process. We show in Figure 3 the result of applying this altered algorithm to the logic program of Example 1.

One can easily adapt the definition of complete labeling for arguments to an analogous concept of complete labeling of conclusions:

Definition 26. *Let* $(\mathcal{A}, \mathcal{S}, f, \mathcal{R})$ *be a CAF and* $\mathcal{L}$ *be a conclusion labeling of* $\mathcal{S}$*. So* $\mathcal{L}$ *is a* complete conclusion labeling *if:*
(i) a conclusion a *is* Out *then for each argument* $A \in f^{-1}(a)$ *there is an argument* B *that attacks* A *with a conclusion* $f(B)$ *labeled* In*;*
(ii) a conclusion a *is* In *then for some argument* $A \in f^{-1}(a)$*, every argument* B *that attacks* A *must have its conclusion* $f(B)$ *labeled* Out.

(iii) a conclusion a is Undecided *then there is no argument $A \in f^{-1}(a)$ for which all arguments B that attack A have $f(B)$ as* Out*; and for some argument $A \in f^{-1}(a)$, there is no argument B that attacks A with conclusion $f(B)$* In*, and there is an argument C that attacks that same A and whose conclusion $f(C)$ is not* Out.

Given the equivalence between the complete labels at the conclusion level and at the arguments level proved by Caminada et al. (2015), we can define the other semantics for the CAFs. From the results of complete semantics for the arguments, it is possible to obtain the equivalent labels at the conclusion level by applying the criteria described in section 3.1. The latter are the complete conclusion labeling solutions and, from them, we can maximize/minimize any desired label. With this, we obtain the other semantics mentioned in this work, which are defined as:

Definition 27. *Assume L is a conclusion labeling.*
L is grounded if it is complete and the conclusion set In *is minimal.*
L is preferred if it is complete and the conclusion set In *is maximal.*
L is stable if it is complete and no conclusion remains Undecided.
L is semi-stable if it is complete and the conclusion set Undecided *is minimal.*

Given that the maximization/minimization process was done directly at the conclusion level, the desired equivalences are obtained:

Theorem 2. *Let $\mathbf{P}$ be a logic program and $\mathbf{C} = (\mathcal{A}, \mathcal{S}, f, \mathcal{R})$ the CAF generated by the modified WCG algorithm. Then the complete, preferred, grounded, stable and semi-stable conclusion labels of $\mathbf{C}$ are identical to the labels assigned respectively by the partial stable, regular, well-founded, stable and L-stable models of $\mathbf{P}$*

Example 4. *(continuing Example 3). Since in CAFs we apply the maximization/minimization of labels directly at the conclusion level, the semi-stable semantics will yield two solutions. Thus it is equivalent to applying the L-stable semantics to the logic program in Example 1.*

Consequently, the semantic correspondence between a logic program and its derived CAF is proved, and it is also possible to demonstrate it in the opposite direction. That is, starting from an CAF and generating a logical program, the correspondences are maintained. We use the following translation process:

Definition 28. *Let $\mathbf{C} = (\mathcal{A}, \mathcal{S}, f, \mathcal{R})$ be a CAF. For each argument A, generate a rule $f(A)$:− $\mathbf{not}$ $f(B_1), ..., \mathbf{not}$ $f(B_m)$ where B_i are the arguments that attack A. We denote $\mathbf{P_C}$ the logic program that consists of the generated rules.*

Example 5. *Applying the algorithm in Definition 28 to the CAF in Figure 3, we get the following set of rules:*

$$r_0 : a :- \mathbf{not}\ c, \qquad r_1 : b :- \mathbf{not}\ a, \mathbf{not}\ b,$$
$$r_2 : c :- \mathbf{not}\ a, \qquad r_3 : b :- \mathbf{not}\ b,$$
$$r_4 : d :- \mathbf{not}\ a.$$

Thus, the following theorem proves the desired equivalences:

Theorem 3. *Let $\mathbf{C} = (\mathcal{A}, \mathcal{S}, f, \mathcal{R})$ be a CAF and $\mathbf{P_C}$ the corresponding logic program . So the partial stable, regular, well-founded, stable, L-stable models of $\mathbf{P}$ assign the same labels to the conclusions as respectively the complete, preferred, grounded, stable, semi-stable semantics of $\mathbf{C}$.*

With this, a correspondence between the semantics of logic programs and argumentation frameworks in both directions is specified. Due to space constraints we refer to our previous work [24] to prove both Theorems 2 and 3.

However, it should be clear that the move to CAFs is not enough to achieve a completely satisfactory one-to-one equivalence with logic programming. That can be seem if we compare the rules of Examples 1 and 5. Rule r_4 differs by the omission/inclusion of the conclusion c in those examples. That means that repeated translations between formalisms loses some information. To solve that problem, in the next section we take the support relations into account.

3.3. Bipolar Conclusion-augmented Argumentation Framework (BCAF)

This section introduces Bipolar Conclusion-augmented Argumentation Frameworks (BCAFs). Most concepts related to bipolarity are based on previous work by Cayrol and Lagasquie-Schiex (2005).

We thus introduce:

Definition 29. *A Bipolar Conclusion-augmented Argumentation Framework (BCAF) is a tuple $(\mathcal{A}, \mathcal{S}, f, \mathcal{R}_-, \mathcal{R}_+)$, with $\mathcal{A}$ being a finite set of arguments, $\mathcal{S}$ a set of conclusions, f being a function of arguments to conclusions, $\mathcal{R}_- \subseteq \mathcal{A} \times \mathcal{A}$ is the attack relation and $\mathcal{R}_+ \subseteq \mathcal{A} \times \mathcal{A}$ is the support relation.*

Each argument $A \in \mathcal{A}$ can be defined as a rule H :− $A_1, \ldots, A_m, \mathbf{not}$ $B_1, \ldots, \mathbf{not}$ B_n. We say A has *conclusion* H, *rules* $\{H$:− $A_1, \ldots, A_m, \mathbf{not}$ $B_1, \ldots, \mathbf{not}$ $B_n\}$, *vulnerabilities* Vul(A) $\{B_1, \ldots, B_n\}$ and *necessities* Nec(A) $\{A_1, \ldots, A_m\}$.

The attack relation is defined in the usual sense of Dung's work, that is, an argument attacked by another that is accepted, must be rejected. In a BCAF, one argument attacks another when its conclusion is one of the other's vulnerabilities. The support relation, for now based on the discussed necessary support and on the work by Alfano et al. (2020), is defined as follows:

Definition 30. *An argument A supports another argument B iff $f(A) = a$ is a necessity of B. Thus, for B to be accepted, at least one of the supports A_i with $f(A_i) = a$ must also be accepted.*

Given the ways of how an argument can relate to another in a BCAF, the definition of a redundant BCAF is relevant:

Definition 31. *Let $AF = (\mathcal{A}, \mathcal{S}, f, \mathcal{R}_-, \mathcal{R}_+)$ be a BCAF. AF is said to be redundant if there is at least pair of arguments $A, B \in \mathcal{A}$ such that $f(A) = f(B)$, Vul(A) = Vul(B) and Nec(A) = Nec(B).*

Also of interest is the definition of the well-formed BCAF as adapted from Dvorák and Woltran (2019):

Definition 32. *Let $AF = (\mathcal{A}, \mathcal{S}, f, \mathcal{R}_-, \mathcal{R}_+)$ be a BCAF. AF is said to be well-formed if all the arguments $A \in \mathcal{A}$ that hold the same conclusion $f(A)$ attack and support the same arguments.*

From this point on we will assume the BCAFs to be well-formed and non-redundant unless stated otherwise.

As previously mentioned, Caminada et al. (2015) studied the WCG algorithm for the translation of logic programs into Dung's argumentation frameworks. Here, this translation is adapted for BCAFs and the semantic equivalences are once again proved. We thus propose an adapted WCG algorithm for BCAFs:

1. Starting with a rule set, process each rule at a time;
2. If a rule of the form $H :- \textbf{not } B_1, \ldots, \textbf{not } B_n$ is found, then generate an argument A with *conclusion* H, *rules* $\{H :- \textbf{not } B_1, \ldots, \textbf{not } B_n\}$ and *vulnerabilities* $\text{Vul}(A)$ $\{B_1, \ldots, B_n\}$;
3. If a rule of the form $H :- A_1, \ldots, A_m, \textbf{not } B_1, \ldots, \textbf{not } B_n$ is found, then generate an argument A with *conclusion* H, *rules* $\{H :- A_1, \ldots, A_m, \textbf{not } B_1, \ldots, \textbf{not } B_n\}$, *vulnerabilities* $\text{Vul}(A)$ $\{B_1, \ldots, B_n\}$ and *necessities* $\text{Nec}(A)$ $\{A_1, \ldots, A_m\}$;
4. After going through all the rules, the relations between arguments are established. If an argument A has a conclusion that is present in the vulnerabilities of another argument B, then A attacks B. On the other hand, if the conclusion of A is present in the necessities of B, then A supports B. With this, and keeping that each argument is linked to a conclusion, the bipolar conclusion-augmented argumentation graph is created.

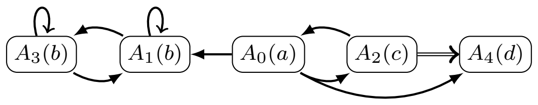

Figure 4: A Bipolar Conclusion-augmented Argumentation Graph

Figure 4 shows a bipolar conclusion-augmented graph, where the adapted WCG algorithm was applied to the rules in Example 1.

One should ask whether, like the original WCG algorithm, the adapted version guarantees the equivalence of the P-stable semantics in logic programming and the complete semantics in the bipolar conclusion-augmented argumentation framework. The answer to this question is once again yes, but to prove it a few definitions must be stated.

First, the rule for translating labels from arguments to conclusions is the same as the one discussed earlier, where each conclusion is represented by the argument that best defends it.

Second, we define the functions that translate labelings from arguments to conclusions and vice versa. These definitions are adapted from Caminada et al. (2015) to the context of BCAFs:

Definition 33. *Let P be a logical program and $AF_P = (\mathcal{A}, \mathcal{S}, f, \mathcal{R}_-, \mathcal{R}_+)$ its associated bipolar conclusion-augmented argumentation structure. Let ArgLabs be the set of all argument labels from AF_P and let ConcLabs be the set of all conclusion labels from P and AF_P*

1. *We define an ArgLab2ConcLab function: ArgLabs $\rightarrow$ ConcLabs such that for each ArgLab $\in$ ArgLabs, it is true that ArgLab2ConcLab(ArgLab) is the associated conclusion labeling of ArgLab;*
2. *We define an ConcLab2ArgLab function: ConcLabs $\rightarrow$ ArgLabs such that for each ConcLab $\in$ ConcLabs and each $A \in \mathcal{A}$ it is true that:*
    a) *ConcLab2ArgLab(ConcLab)(A)* = In *if for each $v \in$ Vul(A) it is true that ConcLab(v) =* Out *and for each $w \in$ Nec(A) it is true that ConcLab(w) =* In*;*
    b) *ConcLab2ArgLab(ConcLab)(A)* = Out *if there is a $v \in$ Vul(A) such that ConcLab(v) =* In *and/or if there is a $w \in$ Nec(A) such that ConcLab(w) =* Out*;*
    c) *ConcLab2ArgLab(ConcLab)(A)* = Undecided *if not for all $v \in$ Vul(A) ConcLab(v) =* Out*; if there is no $v \in$ Vul(A) such that ConcLab(v) =* In*; if it is not true that for all $w \in$ Nec(A) ConcLab(w) =* In *and there is no $w \in$ Nec(A) such that ConcLab(w) =* Out.

Given the definitions above we can state the following theorem, inspired by Theorem 19 from [4]:

Theorem 4. *In the case of complete argument labellings and complete conclusion labellings, the functions ArgLab2ConcLab and ConcLab2ArgLab are bijections and each other's inverse*

Proof. The proof is inspired by the proof of Theorem 19 from [4]. It is enough to prove two things:

1. ConcLab2ArgLab(ArgLab2ConcLab(ArgLab)) = ArgLab.
   Let ArgLab be a complete argument labeling and let A be an argument. Three cases are distinguished.
   a) ArgLab(A) = In. From the fact that ArgLab is a complete labeling, it follows that ArgLab(B) = Out for every attacker B of A and that ArgLab(C) = In for at least one argument C with $f(C) = c$ for each conclusion c that supports A. From the definition of attack it follows that for every $b \in$ Vul(A) and for every argument B with f(B) = b, ArgLab(B) = Out. This then implies that for every $b \in$ Vul(A) it is true that ConcLab2ArgLab(ArgLab2ConcLab(ArgLab))(b) = Out. Similarly, from the definition of support, it follows that for every $c \in$ Nec(A), for at least one argument C with f(C) = c, ArgLab(C) = In, that is, ConcLab2ArgLab(ArgLab2ConcLab(ArgLab))(c) = In. By the definition of ConcLab2ArgLab, we finally obtain that ConcLab2ArgLab(ArgLab2ConcLab(ArgLab))(A) = In;
   b) ArgLab(A) = Out. From the fact that ArgLab is a complete argument labeling, it follows that there is an attacker B of A such that ArgLab(B) = In or there is a set of supporters C of A with the same conclusion c such that ArgLab(C) = Out. Let b = f(B) and c = f(C). From the definition of attack, it follows that $b \in$ Vul(A). From the definition of ArgLab2ConcLab, it follows that ArgLab2ConcLab(ArgLab)(b) = In. Likewise, from the definition of support, it follows that $c \in$ Nec(A) and ArgLab2ConcLab(ArgLab)(c) = Out. From the definition of ConcLab2ArgLab, it follows that ConcLab2ArgLab(ArgLab2ConcLab(ArgLab))(A) = Out;
   c) ArgLab(A) = Undecided. From the fact that ArgLab is a complete argument labeling, it follows that not every attacker C of A has ArgLab(C) = Out and/or not every argument E with $f(E) = e$ for each conclusion e that supports A has ArgLab(E) = In (i). There is also no attacker D of A that has ArgLab(D) = In and/or no conclusion $e \in$ Nec(A) such that all arguments E with $f(E) = e$ have ArgLab(E) = Out (ii). From (i) together with (ii) it follows that there is an attacker B of A with ArgLab(B) = Undecided and/or a group E of arguments with the conclusion $e \in$ Nec(A) with ArgLab(E) = Undecided. Let $b = f(B)$ and $e = f(E)$. From (ii) together with the definition of attack, it follows that there is no argument B' with $f(B') = b$ such that ArgLab(B') = In. So ArgLab2ConcLab(ArgLab)(b) = Undecided. Likewise, (ii) together with the definition of support implies that for some conclusion $e \in$ Nec(A) ArgLab2ConcLab(ArgLab)(e) = Undecided (iii). Furthermore, from (ii) together with the definitions of attack and support, it follows that for each argument D with $f(D) \in$ Vul(A) it is valid that ArgLab(D) $\neq$ In and for at least one argument F with a conclusion $f \in$ Nec(A), ArgLab(F) $\neq$ Out. Therefore, for every $d \in$ Vul(A), ArgLab2ConcLab(ArgLab)(d) $\neq$ In and for at least one $f \in$ Nec(A), ArgLab2ConcLab(ArgLab)(f) $\neq$ Out (iv). Finally, From (iii) and (iv), and the definition of ConcLab2ArgLab, it follows that ConcLab2ArgLab(ArgLab2ConcLab(ArgLab))(A) = Undecided.
2. ArgLab2ConcLab(ConcLab2ArgLab(ConcLab)) = ConcLab.
   Let ConcLab be a complete conclusion labeling. This, by definition, implies that there is a complete labeling of ArgLab arguments with ArgLab2ConcLab(ArgLab) = ConcLab. As noted earlier, it is true that ConcLab2ArgLab(ArgLab2ConcLab(ArgLab)) = ArgLab. It then follows that ConcLab2ArgLab(ConcLab) = ArgLab. This implies that ArgLab2ConcLab(ConcLab2ArgLab(ConcLab)) = ArgLab2ConcLab(ArgLab). Combining these observations, we finally get ArgLab2ConcLab(ConcLab2ArgLab(ConcLab)) = ConcLab.

We thus obtain the desired result. □

Given the proof above, the desired equivalence can be stated:

Theorem 5. *Let* $\mathbf{P}$ *be a logic program and* $AF_P = (\mathcal{A}, \mathcal{S}, f, \mathcal{R}_-, \mathcal{R}_+)$ *the BCAF generated by the modified*

WCG algorithm. Then the complete, preferred, grounded, stable and semi-stable conclusion labels of AF_P are identical to the labels assigned respectively by the partial stable, regular, well-founded, stable and L-stable models of $\mathbf{P}$

Proof. Given the proof of theorem 4, the results obtained by the P-stable semantics in the logic program and by the complete semantics in the BCAF are equivalent. From that, we can prove the equivalences between the preferred, grounded, stable and semi-stable semantics of the BCAF with respectively the regular, well-founded, stable and L-stable semantics from logic programming. This is due to the fact that the latter are special cases of complete/P-stable solutions in which there is a maximization or minimization of a given label at the conclusion level. □

Example 6. *(Continuing Examples 3 and 4.) The results obtained by the complete semantics when applied to the BCAF in Figure 4 are the same as the one shown in Example 3. Since the maximizing/minimizing of labels is done at the conclusion level, the BCAF semi-stable semantics produces the same results as the logic programming L-stable semantics.*

The translation of BCAFs to a logic program is slightly changed from the version shown for CAFs:

Definition 34. *Let* $\mathbf{C} = (\mathcal{A}, \mathcal{S}, f, \mathcal{R}_-, \mathcal{R}_+)$ *be a BCAF. For each argument* A*, a rule is generated as* $f(A) :- f(A_1), ..., f(A_m), \textbf{not } f(B_1), ..., \textbf{not } f(B_n)$ *where* $A_1 ... A_m$ *are the arguments that support* A *and* $B_1 ... B_n$ *are the ones that attack it. We denote as* $\mathbf{P_C}$ *the logical program constructed by this method.*

Interestingly, two-way translations ensure that there is no loss of information, so the logic programs and non-redundant BCAFs generated in repeated translations are always the same. That can be seem if we apply the translation algorithm to the graph in Figure 4. We obtain exactly the rules of Example 1, which in turn are the same rules that were translated to the BCAF. It is thus clear that well-formed and non-redundant BCAFs guarantee correspondences with logic programming in translations in both directions.

Hence we now have the desired one-to-one equivalence between argumentation (well-formed and non-redundant BCAFs) and logic programs.

We also argue that, despite not being a one-to-one equivalence, the results are relevant even for the redundant BCAF case. It is so because even if the BCAF graph generated by multiple translations is not the same, the conclusions and the relations they hold to each other will remain the same. That means that the adapted WCG algorithm eliminates the redundancies, keeps the desired relations between conclusions and maintains the equivalence of semantics.

In short, by introducing the support relation first proposed by Alfano et al. (2020) to well-formed and non-redundant CAFs and proving the equivalence between semantics, we have shown that both formalims yield the same results and can be translated from one to the other without any loss of information.

3.4. Modeling Different Types of Support

As discussed in Section 2.2, there is, in the literature, more than one possible interpretation for the meaning of the support relation in argumentation frameworks. So far, BCAF has been dealt with using a specific interpretation of the support relation which, in the special case where each conclusion is uniquely associated with an argument, converges to the necessary support (For more general cases, the definition changes a little, since in BCAFs support is given from the conclusions and if an argument A is accepted and another A' is not, where $a = f(A) = f(A')$, an argument supported by a can still be accepted).

However it seems natural that the BCAFs should be able to model different types of support. The results by Cayrol and Lagasquie-Schiex (2013) that show that there is a translation between the deductive and the necessary supports and by Polberg and Oren (2014) that deduced the same for the evidential and the necessary supports reinforces the intuition behind the previous sentence. We now show that in addition to the similarity with the necessary support, it is possible to redefine the BCAF in order to encompass other interpretations of support.

Let us take the deductive support as an example. This support means that if an argument A supports B, if A is accepted, B must also be accepted. We can adapt Definition 30 to reflect this different interpretation [8]:

Definition 35. *An argument* A *supports another* B*, if* $f(B) = b$ *is a necessity of* A*. Thus, for* A *to be accepted, at least one of the arguments* B_i *with* $f(B_i) = b$ *must also be accepted.*

Given this, the other definitions for the BCAF remain mostly the same, but a different interpretation of the support relation is modeled. However, this definition only corresponds to the deductive support in the special case that each argument has an unique conclusion. For the more general case, there is a similar divergence. Figure 5 shows the BCAF generated once again from the rules in Example 1.

The same procedure can be run in order to adapt the definition of support to the evidential support: A BCAF that contains the evidential support differentiates between two types of arguments: the *prima-facie* arguments, which do not need any support to be accepted, that is, they do not have a set of *necessities*; and common arguments, which need to be supported by an accepted

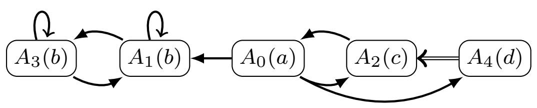

Figure 5: The Bipolar Conclusion-augmented Argumentation Graph generated from the rules in example 1 using Definition 35. It differs from the BCAF in Figure 4 only by the direction of the support.

argument of the first type in order to be accepted as well, that is, they have a set of *necessities.*

Again, the remaining definitions for the BCAF are mostly retained when considering the different support interpretation and, as before, the definition given above is broader than the original definition of evidential support. That is, the two converge only in the special cases where either each conclusion is linked to a single argument and/or no conclusion is linked to both types of arguments.

Example 7. *The graph in Figure 4 is also the representation of the BCAF created from the rules in example 1 using the support definition based on the evidential support. The arguments A_0, A_1, A_2, A_3 are considered prima-facie and A_4 a normal argument*

In general, therefore, it is clear that BCAF have powerful tools for different forms of support, and in special cases, they can correspond exactly to the definitions given for BAFs, and in others, to a more general version of the same principle. Thus, for all interpretations of support, the translation algorithm and the proofs presented can be adapted. That is, regardless of which form the support takes, the properties of well-formed and non-redundant BCAF remain the same, including its one-to-one correspondence to logic programming.

4. Discussion and Related Work

BCAFs have all the advantages that their version without bipolarity (the CAFs) obtains. One gets correspondence between *semi-stable* and *L-stable* semantics and, due to their equivalence with the logic program, BCAFs maintain the same computational complexity as the original model. In addition to that, BCAFs have two additional benefits over CAFs. The first is that the argumentation model now has some form of positive relationship, that is, a form of support. This makes it more expressive (in argumentation terms) than the CAFs and closer to the human way of arguing, as discussed in [7, 13].

The second advantage over CAFs is the one-to-one correspondence with logic programming, i.e. no information is lost in repeated translations between the formalisms (in the well-formed and non-redundant case). In the original model, if we start from an logic program, translate to a CAF and then go back to the logic program, we may lose relationships between the conclusions expressed in the original logic program. With the proposed model and its translations, this no longer happens. This suggests that BCAFs and logic programs can be understood as two different but equivalent formalisms, so that the properties of one can be properly translated to the other.

Despite the several proposals in the literature about possible translations between logic programs and some form of argumentation framework, to the best of our knowledge, ours is the only one that generates an one-to-one translation between the two formalisms, where translation runs in both directions, for some sizeable class of argumentation frameworks — moreover, with the additional degree of argumentative expressiveness provided by the bipolarity.

Alfano et al. (2020) propose methods of translation between logic programs and BAFs that are very similar to those proposed in this text in definitions 30 and 35. However, our BCAFs model places conclusions as the main goal of the argumentative process; this also changes the translations. In addition, Alfano et al. (2020) focus their work on the translations from Dung's argumentation graphs to logic programs, but not in the reverse case. Our model, on the other hand, deals with translations in both directions and guarantees semantic equivalences by focusing on conclusions.

Kawasaki et al. (2019) also propose a similar translation between logic programs (a variation for legal settings called PROLEG) and BAFs in order to develop a system capable of aiding in legal reasoning. The authors however only deal with the translations from PROLEG to BAFs and not the other way around. In addition to that, we feel our approach is more straightforward and encompasses different interpretations of support.

Other notable proposals include various translation schemes that employ Assumption-Based Argumentation Frameworks [10] and are able to prove equivalences to logic programming [9]. A similar proposal was put forth by Pisano et al. (2020), in which the author created the *Arg-tuProlog*, a tool capable of dealing with the $ASPIC^+$ formalism [28] for logic programming and the Dung style AAFs with preferences and weights [27, 29]. None of them obtain both the properties our model introduces.

5. Conclusion

It is well-known that both abstract argumentation frameworks and logic programs capture broad elements of non-monotonic reasoning. It is also well-known that several semantics for abstract argumentation frameworks correspond to semantics for logic programs and vice-versa, but not all popular semantics satisfy this property. In addition, it is also known that conclusion(claim)-augmented arguments lead to a satisfactory set of semantic correspondences; we have rehearsed here those recent results around CAFs through a hopefully simpler approach. And it is well-known that the semantic correspondences between CAFs and normal logic programs are actually based on translations from argumentation frameworks to logic programs and vice-versa. However, these translations are not satisfactory because they are not really "associative" — a translation followed by a back translation does not necessarily get back to the same point. By bringing ideas from bipolar argumentation, namely the recently explored fact that supports can be translated to positive atoms in rules, we obtained the desired correspondences and translations for a sizeable class of frameworks. We thus have that any well-formed and non-redundant BCAF can be readily translated to a normal logic program and any normal logic program can be readily translated to a BCAF, without any informational loss.

Intuitively, a well-formed and non-redundant BCAF *is* a normal logic program, and a normal logic program *is* a BCAF. To summarize the whole reasoning in this paper, we obtained these equivalences by combining existing results on conclusion(claim)-augmented argumentation frameworks and on the translation of supports.

In addition, we showed that our bipolar model allows the support relation to be interpreted in several different ways, converging, in special cases, with the most common definitions found in the literature for bipolar argumentation models.

In future work, BCAFs should be expanded to include uncertainty, and in particular to handle probabilistic argumentation, while maintaining their desirable properties in relation to logic programming. We hope the resulting probabilistic formalism will improve the debate about the meaning of probabilities in argumentation and will provide a solid basis for argumentation algorithms to be implemented for real systems.

Acknowledgments

This work was carried out at the Center for Artificial Intelligence (C4AI-USP), with support by the São Paulo Research Foundation (FAPESP grant 2019/07665-4) and by the IBM Corporation. The first author is partially supported by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) grant 88887.616392/2021-00. The second author was partially supported by CNPq grant 312180/2018-7. We acknowledge support by CAPES - Finance Code 001.

We thank a reviewer who found a technical flaw in our submission, thus leading to this improved version of our paper.

References

[1] K. Atkinson, P. Baroni, M. Giacomin, A. Hunter, H. Prakken, C. Reed, G. Simari, M. Thimm, S. Villata, Towards Artificial Argumentation, AI Magazine 38 (2017) 25–36. URL: https://ojs.aaai.org/index.php/aimagazine/article/view/2704. doi:10.1609/aimag.v38i3.2704.

[2] P. Baroni, F. Toni, B. Verheij, On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games: 25 years later, Argument Computation 11 (2020) 1–14. URL: www.procon.org. doi:10.3233/AAC-200901.

[3] P. M. Dung, On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games, Artificial Intelligence 77 (1995) 321–357.

[4] M. Caminada, S. Sá, J. ao Alcântara, W. Dvorak, On the equivalence between logic programming semantics and argumentation semantics, International Journal of Approximate Reasoning 58 (2015) 87–111.

[5] W. Dvorák, S. Woltran, Complexity of abstract argumentation under a claim-centric view, 33rd AAAI Conference on Artificial Intelligence, AAAI 2019, 31st Innovative Applications of Artificial Intelligence Conference, IAAI 2019 and the 9th AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019 (2019) 2801–2808. doi:10.1609/AAAI.V33I01.33012801.

[6] A. Rapberger, Defining argumentation semantics under a claim-centric view, in: CEUR Workshop Proceedings, volume 2655, CEUR-WS, 2020.

[7] C. Cayrol, M. C. Lagasquie-Schiex, On the acceptability of arguments in bipolar argumentation frameworks, in: L. Godo (Ed.), Symbolic and Quantitative Approaches to Reasoning with Uncertainty, Springer Berlin Heidelberg, Berlin, Heidelberg, 2005, pp. 378–389.

[8] 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. doi:10.1017/S1471068420000253.

[9] M. Caminada, C. Schulz, On the equivalence between assumption-based argumentation and logic programming, Journal of Artificial Intelligence Research 60 (2017) 779–825.

[10] K. Cyras, Q. Heinrich, F. Toni, Computational complexity of flat and generic Assumption-Based Argumentation, with and without probabilities, Artificial

Intelligence 293 (2021) 103449. URL: www.elsevier.com/locate/artint. doi:10.1016/j.artint.2020.103449.

[11] P. Baroni, M. Caminada, M. Giacomin, An introduction to argumentation semantics, The Knowledge Engineering Review (2004) 1–24.

[12] S. Polberg, A. Hunter, Empirical evaluation of abstract argumentation: Supporting the need for bipolar and probabilistic approaches, International Journal of Approximate Reasoning 93 (2018) 487–543. doi:https://doi.org/10.1016/j.ijar.2017.11.009.

[13] C. Cayrol, M.-C. Lagasquie-Schiex, Bipolarity in argumentation graphs: Towards a better understanding, International Journal of Approximate Reasoning 54 (2013) 876–899. Special issue: Uncertainty in Artificial Intelligence and Databases.

[14] A. Gargouri, S. Konieczny, P. Marquis, S. Vesic, On a notion of monotonic support for bipolar argumentation frameworks, in: Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS, volume 1, International Foundation for Autonomous Agents and Multiagent Systems (IFAAMAS), 2021, pp. 546–554.

[15] F. Nouioua, V. Risch, Bipolar argumentation frameworks with specialized supports, in: Proceedings - International Conference on Tools with Artificial Intelligence, ICTAI, volume 1, 2010, pp. 215–218. doi:10.1109/ICTAI.2010.37.

[16] F. Nouioua, V. Risch, Argumentation frameworks with necessities, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 6929 LNAI (2011) 163–176.

[17] G. Boella, D. M. Gabbay, L. Van Der Torre, S. Villata, Support in abstract argumentation, in: Frontiers in Artificial Intelligence and Applications, volume 216, IOS Press, 2010, pp. 111–122.

[18] N. Oren, T. J. Norman, Semantics for evidence-based argumentation, in: Frontiers in Artificial Intelligence and Applications, volume 172, IOS Press, 2008, pp. 276–284.

[19] N. Oren, C. Reed, M. Luck, Moving between argumentation frameworks, Frontiers in Artificial Intelligence and Applications 216 (2010) 379–390.

[20] N. Potyka, Generalizing Complete Semantics to Bipolar Argumentation Frameworks, in: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), volume 12897 LNAI, Springer Science and Business Media Deutschland GmbH, 2021, pp. 130–143. doi:10.1007/978-3-030-86772-0_10.

[21] M. G. Escanuela Gonzalez, M. C. Budan, G. I. Simari, G. R. Simari, Labeled Bipolar Argumentation Frameworks, Journal of Artificial Intelligence Research 70 (2021) 1557–1636. URL: https://jair.org/index.php/jair/article/view/12394. doi:10.1613/JAIR.1.12394.

[22] W. Dvorák, A. Rapberger, S. Woltran, On the relation between claim-augmented argumentation frameworks and collective attacks, in: Frontiers in Artificial Intelligence and Applications, volume 325, IOS Press BV, 2020, pp. 721–728. doi:10.3233/FAIA200159.

[23] W. Dvořák, A. Rapberger, S. Woltran, Argumentation semantics under a claim-centric view: Properties, expressiveness and relation to SETAFs, 17th International Conference on Principles of Knowledge Representation and Reasoning, KR 2020 1 (2020) 340–349. doi:10.24963/KR.2020/35.

[24] V. H. N. Rocha, F. G. Cozman, A credal least undefined stable semantics for probabilistic logic programs and probabilistic argumentation, in: International Conference on Principles of Knowledge Representation and Reasoning (KR2022), to appear.

[25] S. Polberg, N. Oren, Revisiting Support in Abstract Argumentation Systems, Frontiers in Artificial Intelligence and Applications 266 (2014) 369–376. URL: https://ebooks.iospress.nl/doi/10.3233/978-1-61499-436-7-369. doi:10.3233/978-1-61499-436-7-369.

[26] T. Kawasaki, S. Moriguchi, K. Takahashi, Reasoning by a Bipolar Argumentation Framework for PROLEG, in: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), volume 11717 LNAI, Springer, 2019, pp. 115–130. URL: http://link.springer.com/10.1007/978-3-030-31605-1_10. doi:10.1007/978-3-030-31605-1_10.

[27] G. Pisano, R. Calegari, A. Omicini, G. Sartor, ArgtuProlog: A tuProlog-based argumentation framework, in: F. Calimeri, S. Perri, E. Zumpano (Eds.), CILC 2020 – Proceedings of the 35th Italian Conference on Computational Logic, volume 2710 of *CEUR Workshop Proceedings*, CEUR-WS, Aachen, Germany, 2020, pp. 51–66. URL: http://ceur-ws.org/Vol-2710/paper4.pdf.

[28] S. Modgil, H. Prakken, The *ASPIC*$^+$ framework for structured argumentation: a tutorial, Argument and Computation 5 (2014) 31–62.

[29] M. Billi, R. Calegari, G. Contissa, G. Pisano, G. Sartor, G. Sartor, Explainability through argumentation in logic programming, in: CAUSAL'21: Workshop on Causal Reasoning and Explanation in Logic Programming, 2021.