mathematics

MDPI

Article

Formal Bipolar Weighted Semantics for Generalized Argumentation Frameworks

Stefano Ferilli

Department of Computer Science, University of Bari "Aldo Moro", 70100 Bari, Italy; stefano.ferilli@uniba.it

Abstract

Argumentation is the inferential strategy aimed at coping with inconsistent knowledge within a logic theory. Abstract argumentation is based on a graph model where nodes are uninterpreted claims and arcs denote relationships between pairs of claims. Different frameworks provide different expressive power and different strategies ('semantics') to decide which arguments are justified. The Generalized Argumentation Framework was proposed as a unifying model that can simulate major pre-existing frameworks in the literature and combine them into a single, seamless model. It also provides additional power. This paper discusses its formal properties and proposes and investigates different evaluation strategies and semantics that can leverage its specific features and full power.

Keywords: logic in artificial intelligence; knowledge representation; theory of languages and software systems for artificial intelligence; automated reasoning; abstract argumentation; argumentation semantics

MSC: 68T27; 68T30; 68T35

1. Introduction

Argumentation is an inferential strategy for coping with inconsistent knowledge and assigning a justification value to contrasting positions [1]. Each position is supported by a set of claims (the *arguments*) that can attack the arguments supporting competing positions and defend themselves against such attacks. In Artificial Intelligence, argumentation is a branch of Automated Reasoning concerned with developing models, approaches, techniques, and systems for dealing with different aspects of argumentative reasoning. Abstract argumentation, in particular, abstracts away from the internal structure and interpretation of arguments and focuses on their relationships. Attacks form the core of the system. Several independent (Abstract) Argumentation Frameworks (AFs for short) [2] have been defined. They express different kinds and combinations of relationships and associated information and are consequently subject to different expressiveness limitations. *Generalized Argumentation Framework*, or GAFs [3], can simulate many such frameworks, also allowing them to coexist and cooperate and adding new features, resulting in a much more powerful model to carry out abstract argumentation. Among other things, it can express both attacks and supports between arguments, and the 'strength' of each argument and of their relationships.

AFs can be modeled as directed, weighted graphs, where the nodes are the arguments, and the arcs are the relationships between pairs of arguments. A relevant question is how the arguments directly and indirectly affect each other along the paths in the graph, i.e., how the weights in the graph are combined to determine the degree of justification of

Check for updates

Academic Editor: Adolfo Ballester-Bolinches

Received: 7 September 2026
Revised: 4 October 2026
Accepted: 16 October 2026
Published: 26 October 2026

Copyright: © 2026 by the author. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

the arguments. This paper provides a unified account of GAFs, reviews their state of the art, and extends the framework with theoretical and practical/procedural contributions. Specifically, its novel contributions are as follows:

- Providing a unified account of GAFs by collecting and reconciling the partial works available in the literature;
- Extending the GAF framework with concepts and computational procedures for evaluating its arguments while leveraging their specific features and expressive power; the proposed approach also combines aspects of two established approaches to semantics (see next section);
- Studying both the GAF model and its procedures and providing formal results on them.

The paper is organized as follows. After recalling some basics in the next section, Sections 3 and 4 formally define and discuss the proposed strategy. Then, Section 5 discusses related work, and Section 6 illustrates its behavior in practice. Finally, Section 7 concludes the paper.

2. Basics

According to the original Abstract Argumentation setting [2]:

Definition 1. *An* argumentation framework *(*AF *for short) is a pair* $F = \langle \mathcal{A}, \mathcal{R} \rangle$*, where* $\mathcal{A}$ *is a finite set of arguments and* $\mathcal{R} \subseteq \mathcal{A} \times \mathcal{A}$ *is an attack relation (meaning that, given* $\alpha, \beta \in \mathcal{A}$*, if* $\alpha \mathcal{R} \beta$ *then* α *attacks* β*).*

While capturing the core of disagreement, attacks alone can only express *reinstatement* ("attacking the attacker of an argument means defending/supporting that argument"), not explicit agreement between arguments. This setting was extended in various directions to overcome its limitations and increase its expressive power. In the following paragraphs, we introduce the extensions that are most closely related to the topic of this paper.

Bipolar AFs (BAFs) [4] can express both the attack and the support relations, supporting the following combinations:

1. Attacking the supporter of an argument α means attacking α;
2. Supporting the attacker of an argument α means attacking α;
3. Supporting the supporter of an argument α means supporting α.

Weighted AFs (WAFs) [5] can associate attacks (only) with weights expressing their relative strength. The lack of support in WAFs is the reason why BAFs and WAFs cannot be combined. Weights and supports have been combined in [6], through an extension of previous work on graph-based computational strategies for AFs, and in [7], through the proposal of a new framework (the *Bipolar Weighted Argumentation Framework*, or BWAF). BWAFs were a precursor to the framework considered in this paper:

Definition 2. *A* BWAF *is a triplet* $F = \langle \mathcal{A}, \hat{\mathcal{R}}, w_{\hat{\mathcal{R}}} \rangle$*, where* $\mathcal{A}$ *is a finite set of arguments,* $\hat{\mathcal{R}} \subseteq \mathcal{A} \times \mathcal{A}$ *and* $w_{\hat{\mathcal{R}}} : \hat{\mathcal{R}} \to [-1, 0[\cup]0, 1]$ *assigns a weight to each relation instance. Within* $\hat{\mathcal{R}}$*, the attack sub-relation is defined as* $\hat{\mathcal{R}}_{att} = \{ \langle \alpha, \beta \rangle \in \hat{\mathcal{R}} \mid w_{\hat{\mathcal{R}}}(\langle \alpha, \beta \rangle) \in [-1, 0[\}$*, while the support sub-relation is defined as* $\hat{\mathcal{R}}_{sup} = \{ \langle \alpha, \beta \rangle \in \hat{\mathcal{R}} \mid w_{\hat{\mathcal{R}}}(\langle \alpha, \beta \rangle) \in]0, 1] \}$.

By expressing the attack and support notions via the weight sign, BWAFs ensure representational symmetry and allow us to smoothly combine them in mathematical formulas. Weight 0 would mean no relationship between two arguments, so it is not considered.

Trust-affected Bipolar Weighted Argumentation Frameworks (T-BWAFs) [8] further extended BWAFs by allowing one to specify a degree of 'strength' or 'reliability' also for the

arguments (not only for their relationships). Different parameters may determine such a value: e.g., the 'intrinsic strength' of an argument [9] may depend on the certainty degree of its reason [10], the importance of the values it promotes (if any) [11], the reliability of its source [12]. In T-BWAFs, it is determined as the result of several factors, internal to the argument (the authority of the source of the argument and its own confidence in the validity of the argument) or external to it (the trust of a community in the source of the argument).

Generalized Argumentation Frameworks [3] combine all these works (and others as well) in a uniform environment, and further extend them.

Definition 3. *A* Generalized Argumentation Framework *(or* GAF*) is a 4-tuple* $F = \langle \mathcal{A}, \mathcal{S}, w_{\mathcal{A}}, w_{\mathcal{R}} \rangle$*, where*

- $\mathcal{A}$ *is a finite set of arguments;*
- $\mathcal{S}$ *is a system providing external information on the arguments in* $\mathcal{A}$*;*
- $w_{\mathcal{A}} \colon \mathcal{A} \mapsto [0,1]$ *assigns a weight to each argument, to be considered as its intrinsic strength, also based on* $\mathcal{S}$*; and*
- $w_{\mathcal{R}} \colon \mathcal{A} \times \mathcal{A} \mapsto [-1,1]$ *assigns a weight to each pair of arguments, also based on* $\mathcal{S}$*.*

$w_{\mathcal{R}} = 0$ means no (attack or support) relation, just like in BWAFs (where 0 was purposely excluded). Arcs with $w_{\mathcal{R}} = 0$ are ignored when drawing the argumentation graph, and the relationship $\hat{\mathcal{R}}$ of BWAFs can be obtained by taking all the arcs for which $w_{\mathcal{R}} \neq 0$.

For every specific application, the knowledge engineer is in charge of defining $\mathcal{S}$ and its impact on $w_{\mathcal{A}}$ and $w_{\mathcal{R}}$. For the purposes of this paper, we may ignore $\mathcal{S}$.

In WAFs, the weight of an attack can be any number. GAFs instead bound the absolute value of weights to $[0,1]$. Such an interval is commonly used in probability and fuzzy theories and therefore readily interpretable. A value of 1 represents a strength at which the attacking argument 'fully' defeats the attacked one (or the supporting argument 'fully' supports the supported one). Using negative weights for attacks and positive weights for supports has two useful consequences: the sign immediately identifies the kind of relationship, and attacks and supports can be combined through the usual sign rule of multiplication (see Figure 1). In particular, the product of an even number of negative signs is positive; accordingly, a sequence containing an even number of attacks amounts to a defense. Conversely, the product of an odd number of negative signs is negative, and a sequence containing an odd number of attacks amounts to an attack [13]:

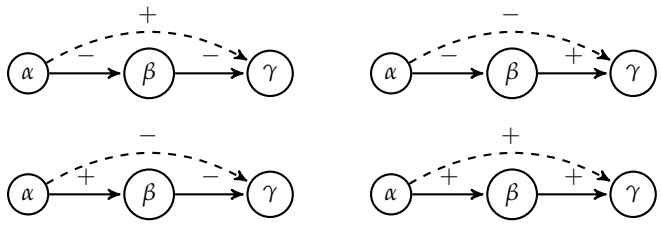
Definition 4. *Given a GAF* $F = \langle \mathcal{A}, \mathcal{S}, w_{\mathcal{A}}, w_{\mathcal{R}} \rangle$ *and a sequence of arguments* $\langle x_0, x_1, \ldots, x_n \rangle$ *such that* $\forall i = 0, \ldots, n : x_i \in \mathcal{A}$*, we say that*

- x_0 g-defends x_n *iff* $\prod_{i=1}^{n} \text{sign}(w_{\mathcal{R}}(x_{i-1}, x_i)) = +1$*;*
- x_0 g-attacks x_n *iff* $\prod_{i=1}^{n} \text{sign}(w_{\mathcal{R}}(x_{i-1}, x_i)) = -1$*.*

The formalization of GAFs also allows us to compute statistics on the direct attacks and supports received by an argument. These statistics can be useful for defining semantics:

Definition 5. *Given a GAF* $F = \langle \mathcal{A}, \mathcal{S}, w_{\mathcal{A}}, w_{\mathcal{R}} \rangle$ *and an argument* $x_0 \in \mathcal{A}$*, we can compute*

- *The number of attacks received by* x_0 *as* $\sum_{x \in \mathcal{A}, w_{\mathcal{R}}(x,x_0)<0} 1$*;*
- *The number of supports received by* x_0 *as* $\sum_{x \in \mathcal{A}, w_{\mathcal{R}}(x,x_0)>0} 1$*;*
- *The* direct justification balance *of* x_0 *as* $\sum_{x \in \mathcal{A}} 1 \cdot \text{sign}(w_{\mathcal{R}}(x, x_0))$*;*
- *The* cumulative weighted attack *received by* x_0 *as* $\sum_{x \in \mathcal{A}, w_{\mathcal{R}}(x,x_0)<0} -w_{\mathcal{R}}(x, x_0)$*;*
- *The* cumulative weighted support *received by* x_0 *as* $\sum_{x \in \mathcal{A}, w_{\mathcal{R}}(x,x_0)>0} w_{\mathcal{R}}(x, x_0)$*;*
- *The* weighted direct justification balance *of* x_0 *as* $\sum_{x \in \mathcal{A}} w_{\mathcal{R}}(x, x_0)$*.*



followed by ×	Support +	Attack −
Support +	Support +	Attack −
Attack −	Attack −	Support +

Figure 1. Sign rule for attacks and supports.

It has been shown that GAFs encompass, combine and extend less expressive models (see [3] for the practical and formal details of how they can be simulated in GAFs). No full evaluation strategy that can leverage the specific features and additional power of the GAF framework has been published so far. Filling this gap is one of the contributions of this paper. In particular, argumentation *semantics* provide a fundamental tool for evaluating AFs. A semantics is a formal evaluation strategy that determines which arguments express a consistent position and/or can be regarded as the 'winners' of the dispute represented by the AF. There are different approaches to semantics [14] in the literature, associated with distinct philosophical and mathematical paradigms: extension-based, labeling-based, and ranking-based semantics.

Extension-based semantics [2] (the traditional approach) evaluate acceptability at a collective level, computing a set of sets of arguments, called 'extensions', each of which represents a coherent, mutually compatible standpoint. The basic, mandatory requirement for an extension is conflict-freeness [2]. Another important notion is defense, meaning that the set collectively counter-attacks outside threats. On top of these notions, the literature has introduced further constraints, and combinations thereof, to define different semantics. Examples include grounded semantics, which provides a unique and highly skeptical viewpoint; preferred semantics, which maximizes the accepted arguments; and stable semantics, which requires every argument outside the extension to be explicitly defeated. The relationships among these semantics have also been extensively studied.

The other two approaches to semantics shift the focus from sets of arguments to individual argument states. Labeling-based semantics map every argument in the graph to one of three discrete evaluation values (*in*, *out*, or *undecided*) by following strict local propagation constraints: an argument is labeled 'in' if and only if all its attackers are 'out', it is labeled 'out' if at least one attacker is 'in', and it is labeled 'undecided' when the status of its attackers is ambiguous or self-contradictory. This approach maintains a strict logical connection to the extension-based paradigm [15]. e.g., a preferred extension corresponds exactly to a preferred labeling, meaning they represent the same underlying logic but offer a different perspective for computation and human interpretation.

Both extension and labeling paradigms share a coarse-grained, all-or-nothing view of defeat, where a single uncountered attack completely invalidates an argument. Ranking-based semantics reject this binary assumption, shifting the paradigm from absolute acceptability to degrees of strength [16]. Instead of producing extensions or discrete labels, they determine a ranking over the entire set of arguments, in the form of a pre-order (reflexive and transitive relation), sorting them from the most acceptable to the weakest. Rather than

relying on defense to completely 'kill' or resurrect an argument, ranking-based semantics treat attacks as a weakening force [17]. An argument's final position in the hierarchy is determined by the structural features considered by the specific ranking-based semantics, such as the total number of direct attacks, the presence of defenders, and the length of propagation chains.

The fundamental differences among these three paradigms lie in their evaluation units, granularity, and treatment of multiple viewpoints. While extension-based and labeling-based semantics may allow multiple valid viewpoints when a graph contains unresolved conflicts, ranking-based semantics typically provide a global ordering or preorder of the arguments. Furthermore, while the classical approaches collapse argument status into discrete categories, ranking-based semantics offer a highly graded, fine-grained evaluation. This makes the classical approaches well suited to binary decision-making and logic programming, whereas ranking-based systems are particularly useful in domains where measuring the relative persuasion level of conflicting opinions is relevant (e.g., online debates or multi-agent systems). The ranking-based paradigm is naturally connected to weighted frameworks; however, proposals to apply weights within binary approaches are also available [18]. As mentioned, the approach proposed in this paper combines aspects of both perspectives, producing a binary decision over acceptability together with a weighted assessment of the reliability of each argument.

3. Weight Spreading Strategy

In an overall approach to abstract argumentation, considering only the immediate attacks on (or supports for) an argument may provide an incomplete basis for determining its justification status or value. AFs are usually represented as directed graphs not only for visualization purposes, but also because graph structure plays a significant role in modeling and analyzing their semantics. In the approach proposed here, the justification of an argument depends on the direct and indirect influences exerted on it by other arguments in the framework. We therefore take these interactions into account.

Let us start by defining two fundamental concepts for our approach:

Definition 6. *Given the argumentation graph G associated with a GAF:*

- *The* justification value propagation *along an arc a from argument α to argument β in G, denoted by* $\mathrm{jvp}(a)$*, is the function computing the amount of justification value that α propagates to β along arc a;*
- *The* justification value *for an argument α in G, denoted by* $\mathrm{jv}(\alpha)$*, is the function that composes the justification contributions associated with all incoming arcs for α in G.*

The $\mathrm{jvp}(\cdot)$ expresses composition 'in depth' (the effect that arguments have on each other through the direct attack or support relationships), while the $\mathrm{jv}(\cdot)$ expresses composition 'in breadth' (how the effects of different relationships converging on the same argument are composed to determine its justification value).

We want the effect of attack/support strengths to be cumulative: the *jv* of a node must depend on all 'previous' nodes and arcs in the graph, and affect the *jv* of all 'subsequent' nodes, where 'previous' and 'subsequent' refer to the existence of directed paths to/from that node. Arguments having no incoming arc get justification equal to their intrinsic strength. Any other argument is assigned a justification value obtained by combining through *jv* the *jvp* values from all of its incoming arcs, which in turn are computed by combining the *jv* of the arc's source node with the arc weight. This has two practical consequences: (1) once the graph, weights, and selected combination functions are fixed, the *jv* of an argument is

uniquely determined; and (2) its computation is also an intermediate step in computing the justification of subsequent arguments in the graph.

Example 1. *As a running example, in the rest of this paper we will use the argumentation graph shown on the left of Figure 2. It consists of 15 nodes, and includes a source cycle that hides node* 3 *(actually, a loop on node* 3*) and a sink cycle that hides node* 14*. Removing the non-interface node* 15 *and its associated arcs* $(14, 15)$ *and* $(15, 14)$*, the residual/stratified graph includes 14 nodes. Its source nodes are* $1, 3, 4$*. Its sink nodes are* $11, 12, 14$*. The internal cyclic part is located in the middle of the graph, involving nodes* $5, 6, 7, 8, 10$*. Of these nodes,* $7, 8, 10$ *get only cyclic contributions.*

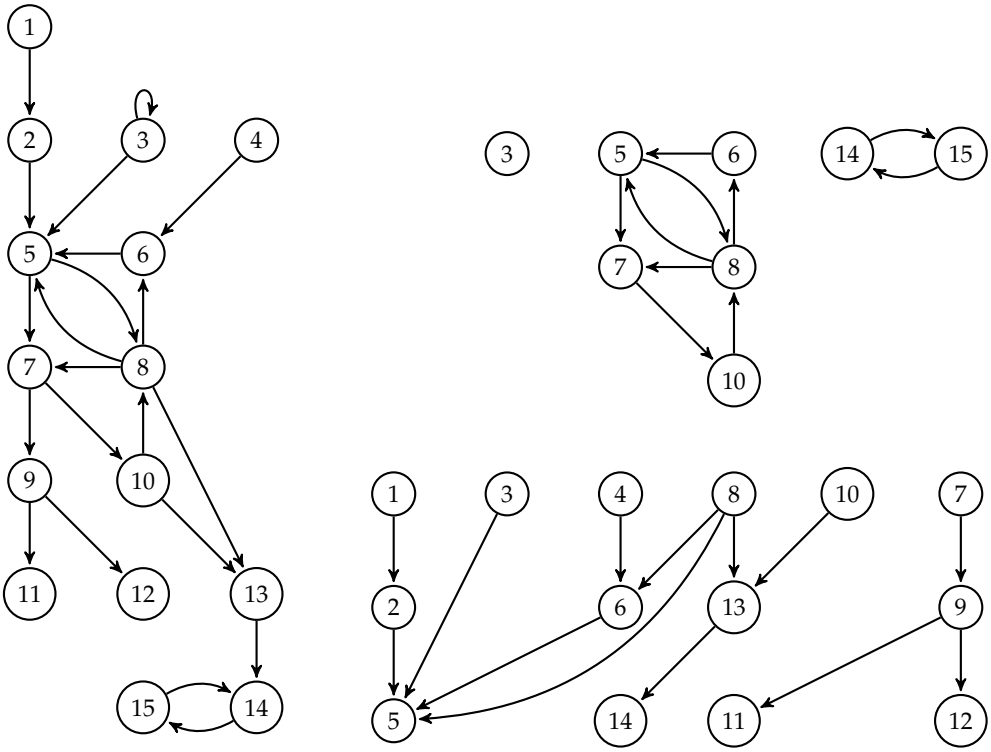


Figure 2. Sample argumentation graph.

3.1. Graph Decomposition Procedure

In general, an argumentation graph may contain many different paths reaching the same argument, as well as cycles. Consequently, a strategy is needed to reduce the evaluation of such complex patterns to the evaluation of single arcs (*jvp*) and composition thereof (*jv*). Such a strategy must tackle the following issues:

- Evaluating independently different paths sharing the same prefix would repeatedly propagate the partial justifications of the prefix, artificially multiplying their contribution to the final result and unnecessarily increasing the computational effort;
- Cycles may cause self-feeding of argumentative contributions for the nodes they involve (especially if traversed several times);
- In a cycle there is no clear notion of what is the start and end of a sequence;
- Variable portions of a cycle might be considered in several paths, multiplying the effect of such portions on the final justification assessment;
- Breaking the cycles by removing any of their arcs would introduce a bias because we would ignore the contribution of the skipped arcs;
- Some cycles may prevent the identification of source or sink nodes in the graph.

To handle them, the strategy described in the next subsections decomposes the original argumentation graph into two independent but complementary components that

together cover the whole graph while avoiding redundancy (that would result in additional computational effort and artificial justification contributions):

- An acyclic structure that captures all strictly linear dependencies; and
- A suitable collection of cycles that captures precisely those argumentative interactions that are not accounted for by the acyclic structure.

3.1.1. Selection of Cycles

The presence of cycles in an argumentation graph gives rise to particularly relevant situations for argumentative reasoning. For cyclic propagation, using all possible cyclic traversals would introduce undesirable redundancy into the justification propagation process. Traversing the same underlying cycle repeatedly would contribute repeatedly to the justification values, artificially reinforcing or weakening them. The selection of cycles is therefore important: we need a subset of cycles that preserves the relevant cyclic information in the original graph while avoiding redundant propagation of justification values. Let us call it the *Cycle Cover (CC)* of an argumentation graph G, and its members the *cyclic components* of G. We take as the CC the set of simple cycles in G (i.e., the cycles having no repeated nodes). By the standard decomposition property of closed walks, every non-empty closed walk in a finite directed graph can be decomposed into a finite multiset of simple cycles [19]. A CC $\mathcal{C}$ also satisfies the following properties:

1. $\mathcal{C}$ is uniquely determined for any graph G;
2. No cycle in $\mathcal{C}$ is reducible;
3. Any arc not used in the acyclic part is used in at least one cycle in $\mathcal{C}$.

Property (1) ensures that the evaluation is deterministic, while (3) guarantees completeness of the evaluation. Property (2) ensures that cycles obtainable as combinations of cycles in the CC are not considered. Recall that our approach is path-based: the cumulative justification contribution reaching a node from different paths is distinct, even when those paths partially overlap. Redundancy must therefore be defined at the level of entire paths, rather than single arcs or subpaths. Every cycle in the CC determines a new, distinct path along which justification propagates, thereby providing a unique justification contribution to the nodes involved. From this perspective, treating only simple cycles as distinct contributions prevents the repeated counting of the same underlying argumentative influence. More precisely, a subpath occurring in the middle of different paths will yield a different contribution in each case, as it is affected by a different initial justification stemming from the preceding portion of the graph. Consequently, the only instance of redundant contribution would involve prefixes—i.e., initial subsequences shared among different paths. However, this is precisely the issue that the progressive evaluation proposed in Section 3.2 addresses, thereby avoiding duplicate processing of such prefixes.

Definition 7. *Let $G = (V, E)$ be a GAF graph and C one of its cyclic components. We define*

- *the external incoming edges of C as $In(C) = \{(u, v) \in E \mid u \notin C,\ v \in C\}$;*
- *the external outgoing edges of C as $Out(C) = \{(u, v) \in E \mid u \in C,\ v \notin C\}$;*

and we call C

- *a* source cycle *iff $In(C) = \emptyset \wedge Out(C) \neq \emptyset$;*
- *a* sink cycle *iff $In(C) \neq \emptyset \wedge Out(C) = \emptyset$;*
- *an* internal cycle *if both $In(C) \neq \emptyset$ and $Out(C) \neq \emptyset$;*
- *an* isolated cycle *if $In(C) = \emptyset$ and $Out(C) = \emptyset$.*

We call source and sink cycles boundary *cyclic components, and* interface nodes *the nodes of the cyclic component that act as contact points to the acyclic part and whose cyclic contribution must be injected.*

Example 2. *The CC for the graph on the left in Figure 2 is shown in the upper-right part of the same figure.* $\langle 3 \rangle$ *is the only source cycle, and* $\langle 14, 15 \rangle$ *is the only sink cycle. All the others are internal cycles (namely:* $C_1 = \langle 5, 8 \rangle$, $C_2 = \langle 5, 8, 6 \rangle$, $C_3 = \langle 7, 10, 8 \rangle$, $C_4 = \langle 5, 7, 10, 8 \rangle$, $C_5 = \langle 5, 7, 10, 8, 6 \rangle$*), and there is no isolated cycle.*

Since distinct connected components of the graph do not interact with each other, throughout the remainder of this work they are considered independently and processed separately. In particular, an isolated cyclic component can be evaluated on its own and does not take part in the construction of the stratified backbone of the other components.

3.1.2. Acyclic Structure

We call *stratified source-sink coverage subgraph* (for short, *stratified subgraph*) of an argumentation graph its acyclic subgraph that provides the 'acyclic backbone' along which justification values are linearly propagated. Let us introduce the following definitions:

Definition 8. *Let* $G = (V, E)$ *be a GAF graph:*

- *a node* $v \in V$ *is a* source node *iff* $\nexists (u, v) \in E$*;*
- *a node* $v \in V$ *is a* sink node *iff* $\nexists (v, u) \in E$*.*

Note that source and sink cycles may hide source or sink nodes (e.g., in Figure 2 source cycle $\langle 3 \rangle$ hides source node 3, and sink cycle $\langle 14, 15 \rangle$ hides sink node 14). Internal cycles, instead, are connected to the remaining graph through both incoming and outgoing arcs to/from interface nodes and therefore do not affect the identification of source and sink nodes:

Proposition 1. *Only source (resp., sink) cycles may hide source (resp., sink) nodes of the stratified backbone.*

Proof. Let C be a cyclic component of G.

If C is a source-cycle, then $In(C) = \emptyset$. Hence no edge from outside C enters C. After separating the internal cyclic edges of C, the remaining interface with the rest of the graph is only through outgoing edges. Thus C behaves as a source component of the residual graph.

Dually, if C is a sink-cycle, then $Out(C) = \emptyset$. Hence no edge from C reaches nodes outside C. After separating the internal cyclic edges of C, the remaining interface with the rest of the graph is only through incoming edges. Thus C behaves as a sink component of the residual graph.

If C is an internal cycle, then both $In(C) \neq \emptyset$ and $Out(C) \neq \emptyset$. Therefore, even after separating its internal cyclic edges, C is connected to the rest of the graph both through incoming and outgoing relationships. Hence it cannot be treated as a source or as a sink component. □

Therefore, to expose the hidden source and sink nodes, source and sink cycles must be separated before the stratified backbone is built, leaving only their interfaces.

Definition 9 (Residual argumentation graph)**.** *The* residual argumentation graph *of an argumentation graph* $G = (V, E)$*, denoted by* $G_R = (V_R, E_R)$*, is obtained from* G *by separating its source, sink, and isolated cyclic components. For each source or sink cyclic component, only its interface nodes and their incoming or outgoing edges are retained. Isolated cyclic components are removed in their entirety.*

https://doi.org/10.3390/math14213901

Then, the stratified subgraph is built layer-wise by progressively adding nodes (and arcs) as long as they become 'resolved' according to the following

Definition 10 (Resolved Node)**.** *During the construction of the stratified subgraph $S = (V_S, E_S)$ of the residual argumentation graph $G_R = (V_R, E_R)$, a node $v \in V_R$ is said to be* resolved *if it is a source node or if, for every incoming edge $(u, v) \in E$, either its contribution has already been determined through cyclic processing or u is already resolved.*
(The justification contribution associated with) an arc is said to be available *if it has already been determined through cyclic processing or if its source node is resolved.*

Once a node is resolved, its justification value can be completely determined. Either it does not depend on the evaluation of any predecessor, or all the required predecessor contributions have already been evaluated during a previous phase of the computation. Once computed, its justification value is final and can immediately contribute to the evaluation of its successors.

Source nodes are resolved because they have no predecessors. Nodes whose incoming edges all appear only in cyclic components of the graph (e.g., node 8 in Figure 2) are also resolved. Their justification value is fully determined by cyclic contributions, which are available before the stratified subgraph is built, and can therefore be computed in advance. We call such nodes *wildcard* nodes.

This leads to the following strategy to build the stratified subgraph:

Layer 0: $L_0 = \{\alpha \in V_R \mid \alpha \text{ is a source or wildcard node in } G_R\}; \quad V_S \leftarrow L_0, \quad E_S \leftarrow \emptyset.$

Layer $i > 0$: $L_i = \{\alpha_k \in V_R \setminus V_S \mid \forall(\alpha_h, \alpha_k) \in E_R : \alpha_h \text{ is already resolved}\};$
$V_S \leftarrow V_S \cup L_i, \quad E_S \leftarrow E_S \cup \{(\alpha_h, \alpha_k) \in E_R \mid \alpha_h \in V_S,\ \alpha_k \in L_i\}.$

By construction, the stratified subgraph is acyclic: every edge added at layer L_i, with $i > 0$, is directed from a node belonging to a preceding layer to a new node in L_i. Hence, layer indices strictly increase along every directed path, making a directed cycle impossible.

Theorem 1 (Completeness of the stratified backbone)**.** *Let $G = (V, E)$ be an argumentation graph and let $G_R = (V_R, E_R)$ be its residual argumentation graph. Then,*

1. *A node is included in a layer only when it is resolved; when a node is included, all its relevant incoming contributions are available: they either belong to cyclic components or come from nodes included in preceding layers;*
2. *Every node of G_R is eventually included in the stratified backbone.*

Proof. By construction, a node is included in a layer only if it is resolved according to Definition 10, and thus it is either:

1. a source node, and therefore has no incoming contributions;
2. a wildcard node, whose incoming contributions are entirely associated with cyclic components and are therefore available from the separate cyclic processing; or
3. a node whose relevant non-cyclic predecessors have already been included in preceding layers; therefore the justification values of all such predecessors are already available by induction on the layer index.

Hence, whenever a node is included, all its relevant incoming contributions are available, either from the cyclic processing or from nodes in preceding layers. This proves (1).

Now assume, by contradiction, that the construction terminates with a non-empty set $U \subseteq V_R$ of nodes still outside the stratified subgraph. Since no node in U is resolved, every node $v \in U$ has at least one relevant incoming dependency from another unresolved node

$u \in U$ that is not handled through a cyclic component. Starting from any node in U and repeatedly following such an incoming dependency yields an infinite sequence of nodes in the finite set U. Therefore, some node must occur more than once, and the corresponding subsequence contains a directed cycle entirely formed by dependencies that have not been handled through cyclic components. Every directed cycle contains a simple cycle. However, all simple cycles of G were identified in the construction of $\mathcal{C}$ before G_R and the stratified subgraph were built. Moreover, removing nodes or edges from G cannot create new cycles. Hence, the simple cycle contained in U was already identified and assigned to the cyclic processing, contradicting the fact that its dependencies remain unhandled.

Therefore, whenever nodes remain outside the stratified subgraph, at least one of them is resolved and can be included in the next layer. Since G_R is finite, the construction terminates after finitely many steps and includes every node. This proves (2). □

Corollary 1 (Partition of the stratified backbone)**.** *Let $L_0, L_1, \ldots, L_k$ be the layers generated by the stratification procedure for the residual argumentation graph $G_R = (V_R, E_R)$. Then the layers form a partition of V_R:*

$$V_R = \bigcup_{i=0}^{k} L_i, \qquad L_i \cap L_j = \emptyset \quad \text{for } i \neq j.$$

Moreover, for every $v \in L_i$, with $i > 0$, every relevant predecessor of v belongs to a preceding layer:

$$\forall u \in \mathrm{Pred}_{G_R}(v), \qquad u \in \bigcup_{h<i} L_h,$$

where $\mathrm{Pred}_{G_R}(v)$ denotes the set of predecessors whose contributions have not already been determined through cyclic processing. Hence, every node is assigned to the earliest layer in which all its relevant incoming contributions are available.

Proof. By Theorem 1, every node of V_R is eventually included in the stratified backbone. Since G_R is finite, the construction terminates after finitely many steps. Hence, for some $k \geq 0$,

$$V_R = \bigcup_{i=0}^{k} L_i.$$

The layers are pairwise disjoint by construction. Indeed, when L_i is generated, only nodes that have not been assigned to any preceding layer are considered. Therefore,

$$L_i \subseteq V_R \setminus \bigcup_{h<i} L_h,$$

and consequently $L_i \cap L_j = \emptyset$ whenever $i \neq j$. Thus, $L_0, \ldots, L_k$ form a partition of V_R.

Concerning the predecessors, let $v \in L_i$, with $i > 0$. Contributions already determined through cyclic processing impose no further dependency on the stratification. Every relevant predecessor of v must therefore be resolved before v can be included and hence belongs to one of the preceding layers. Thus,

$$\mathrm{Pred}_{G_R}(v) \subseteq \bigcup_{h<i} L_h.$$

Finally, we prove the earliest-assignment property. The claim is immediate for nodes in L_0, since source nodes have no incoming contributions, whereas the incoming contributions of wildcard nodes are already available from the cyclic processing. Now let $v \in L_i$, with $i > 0$, and assume, by contradiction, that all its relevant non-cyclic incoming contributions were already available at the beginning of the construction of some layer L_j, with $j < i$.

Then v was already resolved before L_j was constructed. Since, by definition, L_j contains all previously unassigned nodes that are resolved at that stage, v would have been assigned to L_j or to an earlier layer. This contradicts $v \in L_i$, which proves the claim. □

Example 3. *In Figure 2, applying Algorithm 1 to the graph on the left returns the stratified graph on the bottom-right, as follows.*

The source cycle $\langle 3 \rangle$ *is separated, retaining node 3 as its source interface node.*

The sink cycle $\langle 14, 15 \rangle$ *is separated while retaining node 14 as its sink interface node.*

All the other cycles are internal cycles and are handled separately as cyclic components.

Layer 0 is $L_0 = \{1, 3, 4, 7, 8, 10\}$. *Nodes 1 and 4 have no incoming edges and are therefore ordinary source nodes. Node 3 is a source node hidden by the self-cycle* $\langle 3 \rangle$*, previously separated. Nodes 7, 8, and 10 are wildcard nodes because all their incoming edges belong to cyclic components:* $(5, 8)$ *and* $(10, 8)$ *for 8,* $(5, 7)$ *and* $(8, 7)$ *for 7, and* $(7, 10)$ *for 10. Thus, their contribution can be determined from the cyclic part independently of the stratification and before building it.*

Layer 1 is $L_1 = \{2, 6, 9, 13\}$. *Nodes 2 and 9 have only one incoming edge,* $(1, 2)$ *and* $(7, 9)$ *respectively; nodes 6 and 13 have two incoming edges,* $(4, 6)$ *and* $(8, 6)$ *the former, and* $(8, 13)$ *and* $(10, 13)$ *the latter. Since* $\{1, 4, 7, 8, 10\} \subset L_0$*, all of these incoming contributions are available, and thus all of them are resolved.*

Layer 2 is $L_2 = \{5, 11, 12, 14\}$. *The incoming contributions are as follows:* $(2, 5)$*,* $(3, 5)$*,* $(6, 5)$ *and* $(8, 5)$ *for node 5;* $(9, 11)$ *for node 11;* $(9, 12)$ *for node 12; and* $(13, 14)$ *for node 14. All of them are available, their source nodes being in previous layers:* $\{2, 6, 9, 13\} \subset L_1$ *and* $\{3, 8\} \subset L_0$.

We can note that every node occurs just once and is assigned to the earliest layer in which all its relevant incoming contributions are available. The list of arcs not used in the stratified subgraph, along with the cyclic components they belong to, is as follows:

- $(3, 3)$ *to the source cycle* $\langle 3 \rangle$*;*
- $(14, 15)$ *and* $(15, 14)$ *to the sink cycle* $\langle 14, 15 \rangle$*;*
- $(5, 8)$ *to internal cycles* $\langle 5, 8 \rangle$ *and* $\langle 5, 8, 6 \rangle$*;*
- $(5, 7)$ *and* $(10, 8)$ *to internal cycles* $\langle 5, 7, 10, 8 \rangle$ *and* $\langle 5, 7, 10, 8, 6 \rangle$*;*
- $(8, 7)$ *to internal cycle* $\langle 7, 10, 8 \rangle$*;*
- $(7, 10)$ *to internal cycles* $\langle 7, 10, 8 \rangle$*,* $\langle 5, 7, 10, 8 \rangle$ *and* $\langle 5, 7, 10, 8, 6 \rangle$.

Conversely, each cyclic component uses at least one arc from the original graph not present in the stratified subgraph: $\langle 3 \rangle$ *uses* $(3, 3)$*;* $\langle 14, 15 \rangle$ *uses* $(14, 15)$ *and* $(15, 14)$*;* $\langle 5, 8 \rangle$ *and* $\langle 5, 8, 6 \rangle$ *use* $(5, 8)$*;* $\langle 7, 10, 8 \rangle$ *uses* $(8, 7)$ *and* $(7, 10)$*;* $\langle 5, 7, 10, 8 \rangle$ *and* $\langle 5, 7, 10, 8, 6 \rangle$ *use* $(5, 7)$*,* $(7, 10)$ *and* $(10, 8)$.

3.2. Evaluation Procedure

The overall evaluation strategy is shown in Algorithm 1. It returns the set J, which associates each node in the graph with its justification value *jv*. Lines 1–4 initialize J and the set of contributions J_α for each node α. Lines 5–10 then identify the cyclic components and compute and store their contributions to the nodes they involve. For compactness, the result of cumulative applications of *jvp* along the sequence obtained by opening C at α is denoted by $\mathrm{jvp}_\alpha(C)$. Lines 11–15 build $G_R = (V_R, E_R)$, the residual graph of G. Lines 16–32 progressively build the layers L_i $(i = 0, 1, \dots)$ and evaluate $G_S = (V_S, E_S)$, the stratified graph of G_R. The final justification value of each node α_k is computed as soon as all its contributions have been obtained and stored in J_{α_k}; the values in J_{α_k} are then combined. At this point, any nodes not belonging to the stratified graph remain unevaluated. Lines 33–35 evaluate them using only their cyclic contributions, again based on the values in J_α.

Algorithm 1 Justification evaluation strategy for GAFs

Require: $G = (\mathcal{A}, E)$ argumentation graph for GAF $\langle \mathcal{A}, \mathcal{S}, w_{\mathcal{A}}, w_{\mathcal{R}} \rangle$

$J \leftarrow \emptyset$ final justification values
for all $\alpha \in \mathcal{A}$ **do**
  $J_\alpha \leftarrow \emptyset$ {justification contributions for α}
end for
$\mathcal{C} \leftarrow$ set of cyclic components in the graph
for all $C \in \mathcal{C}$ **do**
  for all nodes $\alpha \in C$ **do**
    $J_\alpha \leftarrow J_\alpha \cup \{\mathrm{jvp}_\alpha(C)\}$
  end for
end for
$V_R \leftarrow \mathcal{A}; E_R \leftarrow E$
for all $C \in \mathcal{C}$, C source/sink cycle **do**
  $V_R \leftarrow V_R \setminus \{$ non-interface nodes in $C\}$
  $E_R \leftarrow E_R \setminus \{$ arcs not connected to interface nodes $C\}$
end for
$L_0 \leftarrow \{\alpha \in V_R \mid \alpha$ is a source or wildcard node in $G_R\}$
for all $\alpha \in L_0$ **do**
  compute $\mathrm{jv}(\alpha)$ from J_α
end for
$V_S \leftarrow L_0; E_S \leftarrow \emptyset$
$i \leftarrow 0$
while $V_S \neq V_R$ **do**
  $i \leftarrow i + 1$
  $L_i \leftarrow \{\alpha_k \in V_R \setminus V_S \mid \forall (\alpha_h, \alpha_k) \in E_R,\ \alpha_h$ is already resolved$\}$
  for all $\alpha_k \in L_i$ **do**
    for all $(\alpha_h, \alpha_k) \in E_R$ with $\alpha_h \in V_S$ **do**
      $J_{\alpha_k} \leftarrow J_{\alpha_k} \cup \{\mathrm{jvp}(\alpha_h, \alpha_k)\}$
    end for
    $J \leftarrow J \cup \{\mathrm{jv}(\alpha_k)\}$
  end for
  $V_S \leftarrow V_S \cup L_i; E_S \leftarrow E_S \cup \{(\alpha_h, \alpha_k) \in E_R \mid \alpha_h \in V_S,\ \alpha_k \in L_i\}$
end while
for all $\alpha \in \mathcal{A}, \mathrm{jv}(\alpha) \notin J$ **do**
  $J \leftarrow J \cup \{\mathrm{jv}(\alpha)\}$
end for
return J

Note that the proposed approach allows us to build the stratified graph and compute the justification of arguments at the same time. Each cyclic component is treated separately as a sequence, starting and ending on the same node. Source and sink cycles are handled before constructing the stratified subgraph, thus allowing source and sink nodes to emerge naturally in the acyclic backbone. Internal cycles are evaluated separately and later contribute to the justification values of the corresponding interface nodes through the breadth combination strategy. The evaluation of justification values in the stratified subgraph starts from the source nodes. Further nodes are evaluated as soon as all predecessors contributing to them have been evaluated. This corresponds to proceeding by increasing depth, or layer, in the stratified subgraph. Stratification enables progressive evaluation because, when a node is reached, all its contributions have already been computed and can be combined. The stratified subgraph also allows shared prefixes to be collapsed, so that their partial justification values need to be computed and stored only once.

Example 4. *Let us show a few passages of the computation for the graph in Figure 2. First, the cyclic contributions to nodes belonging to cyclic components are computed, for example:*

- *The justification value of node 3 is obtained by opening it at node 3 the only cycle involving 3 (i.e., the source cycle* $\langle 3 \rangle$*) and computing the* jvp *on the resulting linear path;*
- *The justification value of node 8 is obtained by computing the* jvp *values on the linear paths obtained by all the cyclic components involving 8 (i.e., all internal cycles) considering 8 as the source/sink node to open them, and then applying the* jv *to combine these values.*

Then, the stratified evaluation can start.

Layer 0 *Nodes 1 and 4 get as a justification value their intrinsic strength; nodes 3, 7, 8 and 10 get the justification value computed in the previous cyclic evaluation phase.*

Layer 1 *Node 2 is assigned a justification value by applying the* jvp *to the justification value of node 1 (from the previous layer), arc* $(1, 2)$ *and the strength of node 2. Similarly for the justification value of node 9. Node 6 is assigned a justification value by applying jv (the formula for composition in breadth) to the jvp's associated with all of its incoming arcs, i.e.,* $(4, 6)$ *and* $(8, 6)$*. e.g., for the former, the jvp formula is applied to the current justification value of node 4 (from the previous layer) and arc* $(4, 6)$*. Similarly for node 13.*

Layer 2 *Consider node 5. It is the target of incoming arcs from nodes 2, 3, 6 and 8. So, the jvp is applied to the justification values computed for these nodes in the previous layers and arcs* $(2, 5)$*,* $(3, 5)$*,* $(6, 5)$ *and* $(8, 5)$*, respectively, and then the jv is applied to combine in breadth these 4 values. Similarly for nodes 11 and 12, that have only one incoming arc to consider in the computation. For node 14, the jv must be used to combine the contribution of the acyclic arc* $(13, 14)$ *and the contribution coming from the cyclic evaluation of the sink cycle* $\langle 14, 15 \rangle$*.*

4. Weight Spreading Computation and Combination

Once the graph decomposition and traversal strategy have been fixed, the weight combination functions in depth (*jvp*) and in breadth (*jv*) can be applied. Different functions can be defined, and every combination of an *jvp* and a *jv* will yield a different semantics. In the following subsections, we propose several options and investigate their behavior. Before doing so, we specify the general properties and constraints for the two functions.

In breadth, consider an argument α that receives justification contributions $\mathrm{jvp}(a_i)$ from $n \geq 0$ different arcs $a_i = (\alpha_i, \alpha)$, $i = 1, \ldots, n$ in the argumentation graph. For combining such values in the overall justification value of α we define

$$\mathrm{jv}(\alpha) = \begin{cases} w_{\mathcal{A}}(\alpha) & \text{if } n = 0, \\ \mathrm{cjvp}(\{\mathrm{jvp}(a_i)\}_{i=1,\ldots,n}) \cdot w_{\mathcal{A}}(\alpha) & \text{if } n > 0. \end{cases}$$

where $\mathrm{cjvp}(\cdot)$ denotes a generic combination strategy. The strength of the target node is applied after the justification contributions from the arcs have been combined; it therefore attenuates the overall justification received by the argument. Incorporating this value into the arc-related part of the computation would instead affect the different incoming 'flows' separately. The possibility of associating the target node with a 'strength' that affects the result distinguishes GAFs from many traditional argumentation frameworks. We want *cjvp* to fulfil the following properties:

1. Being upper/lower-bound compliant with respect to the range of values allowed for justification values without needing regularization (which may cause loss of information and unfair behavior);
2. Being commutative and associative (to have a unique, deterministic result, independent of the specific order in which the arcs are processed by the algorithm).

In depth, the quantity of justification propagated through an arc $a = (\alpha, \beta)$ is computed as:

$$\mathrm{jvp}(a) = \mathrm{jvp}(\alpha, \beta) = \mathrm{jvp}(\mathrm{jv}(\alpha), w_{\mathcal{R}}(\alpha, \beta)),$$

where the latter formulation highlights the fact that it depends on the justification value of the source node and the weight of the arc (including its type, attack or support). Note that $\mathrm{jv}(\cdot)$ and $\mathrm{jvp}(\cdot)$ are mutually recursive, with the base case provided by $\mathrm{jv}(\cdot)$ for source nodes in the stratified graph.

We want the composition strategy to fulfil the following properties:

1. Being progressively computable, so that the partial results on the stratified graph can be computed just once and reused for the computation of subsequent nodes in the stratification (this requires the function to be applied only to nodes whose justification value has already been completely defined);
2. Being upper/lower-bound compliant with respect to the range of values allowed for justification values without needing regularization (which may cause loss of information and unfair behavior);
3. Being compliant with the intuition that, while attacks should reduce the justification of the target arguments, supports should increase it.

The semantics defined below express the justification value of an argument as a value in $[-1, +1]$. Its magnitude (absolute value) denotes the amount of justification (in $[0, 1]$), while its sign denotes whether the argument is justified ($+$) or not ($-$) on the basis of topological considerations. Thus, an argument may be defeated while still having a high justification magnitude within the overall position to which it belongs. This numerical representation is compatible with a probabilistic-style interpretation of justification magnitude and, at the same time, makes the argument status (accepted/rejected) immediately identifiable from the sign. $\mathrm{jv}(\alpha) = 0$ is interpreted as total rejection of argument α; consequently, α does not contribute to the computation of the justification value of subsequent nodes in the graph.

4.1. Computation of jv(·)

For the combination in breadth, we propose several options, corresponding to different assumptions about how concurrent justification contributions should interact. In particular, they differ in whether opposite contributions compensate each other, whether one polarity is given precedence, and whether multiple contributions of the same polarity reinforce each other.

Average Justification Value (AJV) is a simple average of the *jvp*'s:

$$\mathrm{cjvp}(\{\mathrm{jvp}(a_i)\}_{i=1,\ldots,n}) = \frac{1}{n}\sum_{i=1}^{n}\mathrm{jvp}(a_i).$$

It is bounded within the justification-value interval, commutative, and progressively computable, provided that the number of values contributing to the current result is also stored. It implements a compensatory behavior: all incoming contributions participate symmetrically in the result, and opposite contributions compensate each other according to their magnitude. As a consequence, the final value is smoothed and lies between the minimum and maximum incoming contributions.

Minimum Justification Value (MinJV)

$$\mathrm{cjvp}(\{\mathrm{jvp}(a_i)\}_{i=1,\ldots,n}) = \min_{i=1,\ldots,n}(\mathrm{jvp}(a_i)).$$

It is bounded, associative and commutative. Semantically, it gives precedence to the least favorable incoming contribution: the presence of a negative contribution determines a negative result whenever it is the minimum value, independently of

stronger positive contributions. Thus, MinJV implements a skeptical, attack-sensitive aggregation policy.

Maximum Justification Value (MaxJV)

$$\text{cjvp}(\{\text{jvp}(a_i)\}_{i=1,\dots,n}) = \max_{i=1,\dots,n}(\text{jvp}(a_i)).$$

It is bounded, associative and commutative. Dually to MinJV, it gives precedence to the most favorable incoming contribution: the presence of a positive contribution determines a positive result whenever it is the maximum value, independently of stronger negative contributions. Thus, MaxJV implements an optimistic, support-sensitive aggregation policy.

Cumulative Justification Value (CJV) implements a reinforcement-sensitive aggregation policy: partial justification values having the same polarity reinforce, rather than smooth, each other, while positive and negative reinforcement are accumulated separately and subsequently opposed. Since justification values are signed, the multisets of positive and negative contributions:

$$P = \{\text{jvp}(a_i) \mid \text{jvp}(a_i) > 0\} \qquad \text{and} \qquad N = \{|\text{jvp}(a_i)| \mid \text{jvp}(a_i) < 0\},$$

are accumulated separately using the following formula:

$$\text{ccjvp}(\{v_1,\dots,v_n\}) = \begin{cases} 0 & \text{if } n = 0, \\ \text{ccjvp}(\{v_1,\dots,v_{n-1}\}) + (1 - \text{ccjvp}(\{v_1,\dots,v_{n-1}\})) \cdot v_n & \text{if } n > 0. \end{cases}$$

and subsequently combined into a single bipolar value as follows:

$$\text{cjvp}(\{\text{jvp}(a_i)\}_{i=1,\dots,n}) = \text{ccjvp}(P) - \text{ccjvp}(N).$$

Positive and negative contributions cumulatively reinforce their respective polarities, while the final sign identifies the prevailing one and the absolute value expresses their cumulative imbalance.

The cumulative function *ccjvp* is bounded in $[0,1]$ by design (at each step only the difference between the current partial value and maximum justification 1 can be added). Therefore, the difference is bounded in $[-1,1]$. As required, *ccjvp* is also commutative and associative. Let $\text{ccjvp}_i = \text{ccjvp}(\{v_1,\dots,v_i\})$. The recurrence can be rewritten as

$$\text{ccjvp}_i = \text{ccjvp}_{i-1} + v_i - \text{ccjvp}_{i-1}v_i = 1 - (1 - \text{ccjvp}_{i-1})(1 - v_i).$$

By iterating, we obtain the closed form

$$\text{ccjvp}(\{v_1,\dots,v_n\}) = 1 - \prod_{i=1}^{n}(1 - v_i),$$

which depends only on the product of the complements $(1 - v_i)$ and is therefore independent of the order and grouping of the contributions. It is progressively computable as long as the *jvp* incoming contribution values are obtained. Within each polarity, it is also monotonically increasing: each additional contribution increases the overall justification value by a proportional portion of the remaining 'justification allowance'. Under repeated positive contributions, it approaches 1 (the upper bound for justification values), which is reached only if some partial justification value is exactly 1.

Another obvious statistic is sum:

$$\text{cjvp}(\{\text{jvp}(a_i)\}_{i=1,\ldots,n}) = \sum_{i=1}^{n} \text{jvp}(a_i),$$

but it is not upper bounded and would therefore require regularization.

The four alternatives therefore express different semantic attitudes toward concurrent evidence. AJV is compensatory and treats positive and negative contributions symmetrically; MinJV and MaxJV are selective, respectively giving precedence to the least and most favorable contribution; CJV is compensatory between opposite polarities but reinforcement-sensitive within each polarity. For instance, two contributions of equal magnitude and opposite sign, $+x$ and $-x$, cancel each other under both AJV and CJV, whereas MinJV returns $-x$ and MaxJV returns $+x$. Conversely, adding further contributions of the same polarity produces explicit cumulative reinforcement under CJV, while MinJV and MaxJV remain determined by the relevant extreme and AJV remains an averaging, rather than cumulative, operator. Thus, different justification values, and possibly different justification statuses, are an expected consequence of the different aggregation principles embodied by these semantics rather than an incidental numerical effect.

4.2. Computation of jvp(·)

In the following, for the sake of compactness we will denote by $p = w_{\mathcal{R}}(\beta, \alpha) \in [-1, 0) \cup (0, +1]$ the strength and type of the relationship/arc $a = (\beta, \alpha)$ between arguments/nodes α and β in G. As said, attacks and supports must be treated differently because they have opposite dialectical effects: support preserves the polarity of the propagated justification, whereas attack reverses it. Besides this common polarity behavior, different propagation functions may embody different assumptions about how the magnitude of an influence evolves along a sequence of relationships.

4.2.1. Supports

For supports we propose only one, cumulative approach, based on the reinforcement principle that a support preserves the dialectical polarity of the propagated justification while increasing its magnitude:

$$\text{jvp}(a) = \text{sign}(\text{jv}(\beta)) \cdot (|\text{jv}(\beta)| + (1 - |\text{jv}(\beta)|) \cdot p).$$

Proposition 2. *Support propagation $jvp(a)$ can be interpreted as a convex interpolation between $jv(\beta)$ and the extreme of the same polarity. Let $l = jv(\beta)$.*

Proof. Since $l = \text{sign}(l)|l|$, we have

$$\text{sign}(l)(|l| + (1 - |l|)p) = l + \text{sign}(l)(1 - |l|)p,$$

and thus:

$$\text{jvp}(a) = \text{jv}(\beta) + \text{sign}(\text{jv}(\beta))(1 - |\text{jv}(\beta)|)p,$$

or, since $\text{sign}(l)(1 - |l|) = \text{sign}(l) - l$:

$$\text{jvp}(a) = (1 - p)\text{jv}(\beta) + p\,\text{sign}(\text{jv}(\beta)).$$

□

If $\text{jv}(\beta) > 0$, the result belongs to $[\text{jv}(\beta), 1]$; if $\text{jv}(\beta) < 0$, it belongs to $[-1, \text{jv}(\beta)]$. This makes immediate the intuition: $p = 0 \Rightarrow$ no reinforcement, $p = 1 \Rightarrow$ reinforcement up to

the extreme of the same polarity. Hence, support propagation preserves dialectical polarity while increasing justification magnitude.

Given the *jv* for the source node, the support will increase the value for the sink node taking a portion of the remaining allowance proportional to the strength of the support. Of course, supporting an already fully justified claim does not add to its justification value. This function is clearly upper-bound compliant. It monotonically increases the justification magnitude as long as a sequence of support relationships is traversed. The increase is proportional to the remaining justification allowance, so that the value asymptotically approaches the upper bound without exceeding it.

4.2.2. Attacks

For attacks, we propose several options. All of them reverse the dialectical polarity of the propagated justification, thus preserving the traditional alternation between attack and reinstatement along attack chains, but they differ in how the magnitude of the propagated influence is transformed.

Local-Complement (LC) locally weakens the justification magnitude propagated by the preceding argument based on the attack strength, and then complements it:

$$\mathrm{jvp}(a) = -\mathrm{sign}(\mathrm{jv}(\beta)) \cdot (1 - |\mathrm{jv}(\beta)| \cdot |p|).$$

In sequences of attacks, the justification value alternates between relatively low and relatively high values according to the parity of the attack chain: an attack weakens the current justification, while an attack against an attacker restores part of the lost justification. Thus, LC interprets attack strength locally: each attack transforms the justification received from the immediately preceding argument, while reinstatement emerges from the repeated reversal of polarity and complementation of magnitude.

Simple Algebraic (SA) simply multiplies the justification propagated by the preceding argument and the attack weight:

$$\mathrm{jvp}(a) = \mathrm{jv}(\beta)p.$$

The attack weight, being negative, naturally encodes the dialectical polarity of the result. Sequences of attacks jointly and progressively reduce the absolute value of the justification: the longer the sequence, the more it approaches 0 (provided that not all weights are 1). Thus, SA embodies an attenuation principle according to which the strength of an indirect influence decreases multiplicatively along an attack chain.

Complemented Residual-Product (CRP) computes the interaction intensity from residual factors $(1 - |p|)$:

$$\mathrm{jvp}(a) = -\mathrm{sign}(\mathrm{jv}(\beta)) \cdot \begin{cases} (1 - |\mathrm{jv}(\beta)|(1 - |p|)) & \text{if } \mathrm{jv}(\beta) > 0 \text{ (accepted)}, \\ (1 - |\mathrm{jv}(\beta)|)(1 - |p|) & \text{if } \mathrm{jv}(\beta) < 0 \text{ (defeated)}; \end{cases}$$

i.e., considering the sign of $\mathrm{jv}(\beta)$,

$$\mathrm{jvp}(a) = \begin{cases} -(1 - \mathrm{jv}(\beta)(1 - |p|)) & \text{if } \mathrm{jv}(\beta) > 0 \text{ (accepted)}, \\ (1 + \mathrm{jv}(\beta))(1 - |p|) & \text{if } \mathrm{jv}(\beta) < 0 \text{ (defeated)}. \end{cases}$$

This option adopts the same cumulation approach based on residual factors as for supports, but alternates the cumulative value and its complement based on current polarity. As a result, it has a restorative behavior: successive attacks may progres-

sively reinstate a defended argument, driving its justification magnitude toward the corresponding extreme. Since the residual product rapidly approaches zero, long attack chains quickly converge towards either 0 or 1, producing a rather strong saturation effect.

To avoid excessively fast saturation, a parameter $\lambda \in (0, 1]$ may be introduced, replacing $|p|$ by $\lambda|p|$. The case $\lambda = 1$ yields the strongest version, whereas $\lambda < 1$ produces a more conservative behavior. This version of CRP is progressively computable and bounded: each application produces a justification value in $[-1, 1]$. Along a homogeneous sequence of attacks, its magnitude alternates according to the parity of the sequence, with the two parity subsequences moving monotonically towards opposite extremes.

Signed Residual-Product (SRP) computes the interaction intensity from residual factors and explicitly represents the dialectical polarity through the sign:

$$\mathrm{jvp}(a) = -\mathrm{jv}(\beta)(1 - |p|).$$

It also relies on residual factors. This results in an attenuating behavior: in a sequence, every attack reduces the amount of justification propagated, while the sign alone records whether the resulting position is attacking or defending.

Complemented Direct-Product (CDP) adopts the same strategy as CRP, but uses the attack strengths instead of their complements:

$$\mathrm{jvp}(a) = -\mathrm{sign}(\mathrm{jv}(\beta)) \cdot \begin{cases} (1 - |\mathrm{jv}(\beta)|)|p| & \text{if } \mathrm{jv}(\beta) > 0 \text{ (accepted)}, \\ (1 - |\mathrm{jv}(\beta)||p|) & \text{if } \mathrm{jv}(\beta) < 0 \text{ (defeated)}; \end{cases}$$

i.e., considering the sign of $\mathrm{jv}(\beta)$,

$$\mathrm{jvp}(a) = \begin{cases} -(1 - \mathrm{jv}(\beta))|p| & \text{if } \mathrm{jv}(\beta) > 0 \text{ (accepted)}, \\ 1 + \mathrm{jv}(\beta)|p| & \text{if } \mathrm{jv}(\beta) < 0 \text{ (defeated)}. \end{cases}$$

Sequences of attacks produce an alternation between low and high magnitudes but using the attack strengths $|p|$ directly.

All options preserve the traditional dialectical behavior of attack and reinstatement: repeated applications alternate the polarity along a sequence. They differ, however, in the principle governing the magnitude of the propagated influence. Only *SA* and *SRP* simultaneously capture both the intensity of the interaction and its dialectical effect using only a product; in all the others, this is obtained by explicitly separating the dialectical polarity from the interaction strength. Table 1 shows at-a-glance a comparison of some relevant features for the various options, where *parity* indicates different formulas depending on the parity of sequences, *residual* means that residual factors are used instead of the direct attack weights, *oscillating* means that the magnitude of justification along sequences of attacks is non-monotonic, *extreme* tells whether the absolute value tends to the extremes (0 or 1 for oscillating options, 0 for monotonic ones), and *fast* specifies if there is a rapid saturation of the value toward the extreme. The latter two parameters are clearly qualitative.

Table 1. Comparison of attack options features.

Option	Parity	Residual	Oscillating	Extreme	Fast
LC			X		X
SA				X	
CRP	X	X	X	X	X
SRP		X		X	
CDP	X		X	X	X

From a conceptual viewpoint, we may say the following.

LC provides a local interpretation of propagation. Each attack locally weakens the current justification, while attacking an attacker locally restores it. Consequently, the justification value evolves as a continuous dialogue between attacks and defenses, closely matching the classical intuition underlying Dung's notion of reinstatement.

SA provides a purely algebraic interpretation. A single mathematical operation, multiplication, simultaneously captures both essential aspects of argument propagation (dialectical polarity and strength of the interaction). No explicit distinction between attack and defense is required; reinstatement emerges automatically.

CRP provides a residual-based interpretation. By explicitly separating polarity from intensity, it distinguishes two concepts that are implicitly combined in other approaches. The resulting behavior models successive attacks as progressively reinstating a defended argument.

SRP retains the use of residual factors from CRP while representing the dialectical polarity directly through the sign of the justification value. This produces a particularly clean interpretation.

CDP combines polarity-conditioned complementation with the direct attack strengths, rather than their residuals. It therefore retains the parity-sensitive behavior of CRP while reversing the role played by strong and weak attack weights in driving the propagated magnitude towards the extremes.

The five options exhibit three distinct patterns of magnitude propagation. SA and SRP form an attenuation pair: both produce a non-increasing magnitude, but SA retains the fraction $|p|$ at each step, whereas SRP retains $1 - |p|$. Therefore, SA preserves more magnitude for strong attacks and SRP for weak attacks, with the two having the same attenuation factor when $|p| = 0.5$. CRP and CDP instead form a parity-sensitive pair. For a positive initial value, as the corresponding cumulative product decreases, CRP drives odd-length sequences towards magnitude 1 and even-length sequences towards 0, whereas CDP produces the opposite pattern. Their relative rate of saturation depends on attack strength: CRP is more extreme when $|p| > 0.5$, while CDP is more extreme when $|p| < 0.5$; the two use the same factor when $|p| = 0.5$. LC does not exhibit either of these general patterns. Its propagated magnitude may increase or decrease at each step according to the current value and the strength of the next attack.

Overall, the five functions should not be regarded as minor mathematical variants of each other. Rather, they represent distinct philosophies of argumentative propagation. We can distinguish three families:

Local propagation. LC models attack propagation as a local transformation of the value received from the preceding argument. Thus, attack and reinstatement emerge from the repeated local alternation between a value and its complement. Unlike the product-based options, the result depends on the ordered sequence of local

transformations. This interpretation is probably the closest to the intuitive idea of a justification value being propagated along a sequence of attacks.

Purely algebraic propagation. SA and SRP rely exclusively on signed multiplication. Their most remarkable feature is that reinstatement does not need to be explicitly modeled through the parity of the attack sequence: it is an immediate consequence of the multiplication of signed quantities. This makes them a mathematically simpler and more elegant formulation.

Polarity-conditioned complementary propagation. CRP and CDP explicitly select different magnitude transformations according to the polarity of the preceding justification value. In both cases, each attack reverses the polarity, while multiplication and complementation are alternated to model the transition between attack and reinstatement. These formulations trade part of the algebraic simplicity for greater modeling flexibility, since the notions of attack, defense and interaction strength can be independently controlled.

4.3. Computational Complexity Evaluation

As to the computational complexity, let us first note that the proposed procedure is deterministic: the simple cycles in the graph are uniquely determined, and after removing them, the stratification procedure is deterministic by construction.

The optimal algorithm for finding and listing all simple cycles [20] reports a time complexity of $\mathcal{O}((V+E)\cdot(C+1))$, and a space complexity of $\mathcal{O}(V+E)$, where C is the total number of simple cycles in the graph (i.e., it spends a linear time $\mathcal{O}(V+E)$ for each cycle found). Then, the iterative algorithm for finding the stratification of a DAG corresponds to the *Longest-Path Layering* algorithm, whose logic originates from early critical path scheduling methods and was adapted to graph visualization [21]. The optimal computational complexity to achieve this stratification is linear ($\mathcal{O}(V+E)$) in time and space (every level-processing step in the algorithm is linear), provided that the algorithm avoids redundant scans of the graph, which can be achieved by adapting the algorithm for topological sorting in [22]. Finally, for each sequence, the computation is linear (also considering that each cycle is expanded into as many sequences as its number of nodes, by opening it at each node). The prefix-aware approach reduces this cost by computing each prefix once for all sequences that share it.

5. Related Work

This work builds on previous publications while extending them in several directions. Reference [7] introduced a jointly bipolar and weighted framework (BWAF), establishing a representation in which the kind of relationship (attack or support) is expressed by the sign of its weight. This representation is inherited by GAFs. Reference [8] extended BWAFs into T-BWAFs by introducing weights for nodes as well as arcs. This feature is also inherited by GAFs. Reference [23] proposed a matrix-based approach for computing some existing semantics in BWAFs and introduced additional matrix-based semantics. Since BWAFs are embedded in GAFs, these semantics can also be considered in the GAF setting; however, they differ from the semantics proposed in this paper. Reference [3] introduced the theoretical GAF framework and showed how some existing frameworks and semantics can be simulated within it. It did not, however, provide GAF-specific semantics, evaluation strategies or computational procedures. Reference [13] is the work most closely related to the present paper and contains the initial ideas revised and extended here. It introduced justification propagation in depth and in breadth and proposed one function for propagating supports in depth, one for propagating attacks in depth, and three basic statistic-related functions for combining justification values in breadth. Its analysis was

limited to sequential patterns and did not consider weighted arguments. The present paper introduces several depth functions for attacks and a more elaborate breadth-combination function. It also develops an evaluation strategy for general graphs, including cycles, and incorporates weighted arguments.

Because adjacency matrices provide one representation of graphs, linear algebra, and matrix theory in particular, have also been used for argument-graph computation. Several works have investigated matrix representations and computations for argumentation frameworks. References [24,25] show how AFs and WAFs can be represented by a matrix, and the basic extensions of an AF can be determined by sub-blocks of its matrix. Reference [26] introduced a matrix-based mathematical approach for determining whether a set of arguments is an extension. Reference [27] proposed a Boolean matrix approach to encode Dung's acceptability semantics, which can be solved by Boolean constraint solvers. Reference [28] investigated the relationship between formal argumentation semantics and measures from social networking theory by considering matrix exponentials. Reference [23] used a matrix-based approach to analyze extension-based semantics, extended these ideas to bipolar and weighted AFs and to ranking semantics, and considered techniques for computing extensions. Since BWAFs are embedded in GAFs, the corresponding results from Reference [23] can also be applied within the GAF setting.

General principles for characterizing gradual and weighted bipolar argumentation semantics have been extensively investigated in the literature. In particular, axiomatic approaches dealt with properties such as directionality, stability, neutrality, monotonicity, reinforcement, and different forms of balance between attacks and supports [6,29]. Modular approaches have further emphasized the usefulness of separating the aggregation of incoming influences from the way such influences affect an argument, and have studied properties such as monotonicity and duality for these components [30]. Recent work confirms the continuing relevance of a principle-based analysis of bipolar gradual semantics by investigating their behavior when uncertainty is associated with the initial argument weights [31]. These principles provide a useful reference for interpreting the semantics proposed in this paper, but they cannot be transferred verbatim to our setting. In GAFs, both arguments and relationships may be weighted, attack and support are uniformly represented by the sign of the relationship weight, and justification values belong to $[-1, 1]$, with the sign representing the dialectical status and the absolute value representing its justification magnitude. Moreover, the proposed evaluation explicitly distinguishes propagation in depth from combination in breadth. This separates the choice of how an attack or support propagates along a path from the breadth-level choice of how concurrent influences are aggregated at their target. So, while the alternative functions proposed in this paper can be related to these established principles, their specific interpretation must also take into account the additional representational and computational features of GAFs.

At the breadth-combination level, some of the proposed alternatives can be directly related to established principles for bipolar semantics. In particular, the symmetric compensation implemented by AJV and CJV is related to the balance intuition underlying the Franklin and Strict Franklin principles [6]: contributions of equal magnitude and opposite polarity cancel each other. MinJV and MaxJV deliberately depart from such symmetric compensation, giving precedence to the negative and positive polarity, respectively. CJV additionally captures a reinforcement principle within each polarity, since further concordant contributions cumulatively increase the corresponding positive or negative component. These differences make explicit that the proposed alternatives are not intended as equivalent approximations of a single ideal semantics, but as different aggregation policies suitable for different assumptions about compensation, polarity preference, and reinforcement.

At the depth-propagation level, all the proposed alternatives preserve the basic dialectical principle of attack and reinstatement: an attack reverses the polarity of the propagated justification, so that an even number of attacks restores the original polarity. Their distinction is about the evolution of the magnitude of indirect influence. LC adopts a local transformation policy, SA and SRP implement multiplicative attenuation, and CRP and CDP implement polarity-conditioned complementary propagation. Thus, the alternatives agree on the qualitative dialectical effect of attack chains while making different assumptions about persistence, attenuation, and saturation of their quantitative influence.

The distinction between depth propagation and breadth combination is related to, but different from, the modular view of bipolar weighted semantics [30]. Modular semantics separate an aggregation function, which summarizes the strengths of the attackers and supporters of an argument, from an influence function, which combines the aggregated influence with the argument's initial weight. In the present approach, instead, *jvp* propagates a signed justification contribution through each weighted relationship, whereas *jv* combines the contributions reaching the same argument. This separation is embedded in a graph-evaluation procedure in which indirect influences are propagated through the stratified backbone and cyclic contributions are processed separately. Thus, depth and breadth are independent dimensions of the semantics: a propagation policy and an aggregation policy can be selected separately and then combined.

The treatment of cycles also distinguishes the proposed approach from recent semantics for weighted bipolar argumentation. Thieyre et al. [31] restrict their analysis of state-of-the-art bipolar gradual semantics to acyclic graphs, noting that several such semantics are not defined or do not converge on arbitrary cyclic graphs. Restricting the analysis to acyclic graphs can, however, exclude relevant argumentation structures. More recently, Wang and Shen [32] address this limitation by defining convergent gradual semantics whose values are obtained through an iterative evaluation process. Our strategy takes a different route: cyclic dependencies are explicitly identified and evaluated as cyclic components, while the remaining dependencies are organized into an acyclic stratified backbone on which justification can be propagated progressively. Hence, in our framework, cycles are handled structurally rather than through global iterative convergence.

6. Behavior Analysis Through Illustrative Examples

We now provide examples of the proposed approach, or parts thereof, under different conditions in order to illustrate its behavior in practice.

6.1. Linear Case

In this section, we analyze the behavior of the *jvp* on sequences of arguments as a simple and intuitive pattern of combination. Consider a path in the argumentation graph, corresponding to a sequence of arguments $P = \langle \alpha_0, \ldots, \alpha_k \rangle$ with homogeneous relationships (i.e., all attacks or all supports):

$$\forall i = 1, \ldots, k : p_i > 0 \vee \forall i = 1, \ldots, k : p_i < 0,$$

where $p_i = w_{\mathcal{R}}(\alpha_{i-1}, \alpha_i), \quad i = 1, \ldots, k$. Note that the computation of *jv* for all nodes in the sequence involves only one incoming arc, and so all different options collapse, computing the same value (that is, the value associated with that arc).

To analyze the 'pure' contribution of attack/support strengths, independent of the intrinsic strength of the arguments, let us first consider the case when arguments are not weighted (i.e., have intrinsic strength equal to 1.0):

$$\forall i = 1, \ldots, k : w_{\mathcal{A}}(\alpha_i) = 1.$$

Note that this does not imply that $\mathrm{jv}(\alpha_0) = 1$. Even when all arguments in the sequence have intrinsic strength 1, argument α_0 may have received contributions from preceding portions of the graph, and its justification value may therefore be any value in $[-1, 1]$.

Example 5. *Given* $jv(\alpha_0) = 0.9$ *and the following sequence of arc weights:*

$$p_1 = 0.8, \quad p_2 = 0.7, \quad p_3 = 0.9, \quad p_4 = 0.8, \quad p_5 = 0.6,$$

the jv magnitudes obtained after processing each arc in the sequence using the different jvp options are as follows (the sign is reported in the last row):

	p_1	p_2	p_3	p_4	p_5
Supports	*0.98*	*0.994*	*0.9994*	*0.99988*	*0.99995*
LC	*0.28*	*0.804*	*0.2764*	*0.77888*	*0.53267*
SA	*0.72*	*0.504*	*0.4536*	*0.36288*	*0.21773*
CRP	*0.82*	*0.054*	*0.9946*	*0.00108*	*0.99957*
SRP	*0.18*	*0.054*	*0.0054*	*0.00108*	*0.00043*
CDP	*0.08*	*0.944*	*0.0504*	*0.95968*	*0.02419*
Attacks effect	$-$	$+$	$-$	$+$	$-$

We observe the following.

Support propagation *monotonically reinforces the initial justification, rapidly approaching the upper bound* 1 *without exceeding it.*

LC *produces a non-monotonic alternation between lower and higher justification magnitudes. This behavior results from repeatedly complementing the portion transmitted by the current attack; it does not, in general, entail a monotonic attenuation of the magnitude.*

SA *monotonically attenuates the propagated magnitude towards zero. Each attack multiplies the current magnitude by its strength, so that the cumulative attenuation is determined by the product of the attack strengths.*

CRP *exhibits a pronounced parity-dependent behavior. Odd-length attack sequences produce negative values whose magnitudes rapidly approach* 1*, whereas even-length sequences produce positive values whose magnitudes rapidly approach* 0*. Hence, long even-length sequences tend to produce an almost complete indirect defense.*

SRP *also alternates polarity, but its magnitude decreases monotonically towards zero through the product of the residual attack factors. At even steps, its magnitude coincides with that produced by CRP, whereas at odd steps the two magnitudes are complementary.*

CDP *displays the parity pattern complementary to that of CRP: odd-length sequences produce negative values whose magnitudes tend towards* 0*, whereas even-length sequences produce positive values whose magnitudes tend towards* 1*. In this example, CDP also approaches the extremes rapidly; therefore, it cannot be claimed that it generally avoids saturation towards* 0 *and* 1*.*

Now, let us analyze how introducing intrinsic argument strengths changes the options' behavior.

Example 6. *Given* $\mathrm{jv}(\alpha_0) = 0.9$*, the same sequence of arc weights as in Example 5:*

$$p_1 = 0.8, \quad p_2 = 0.7, \quad p_3 = 0.9, \quad p_4 = 0.8, \quad p_5 = 0.6,$$

and the following sequence of node strengths:

$$s_1 = 0.8, \quad s_2 = 0.7, \quad s_3 = 0.6, \quad s_4 = 0.5, \quad s_5 = 0.9,$$

the intermediate jv *values and final* jvp *value are as follows:*

	p_1	p_2	p_3	p_4	p_5
Supports	*0.784*	*0.65464*	*0.57928*	*0.45793*	*0.70485*
LC	*0.224*	*0.59024*	*0.28127*	*0.38749*	*0.69075*
SA	*0.576*	*0.28224*	*0.15241*	*0.06096*	*0.03292*
CRP	*0.656*	*0.07224*	*0.59567*	*0.04043*	*0.88544*
SRP	*0.144*	*0.03024*	*0.00181*	*0.00018*	*0.00007*
CDP	*0.064*	*0.66864*	*0.17893*	*0.42843*	*0.30865*
Attacks effect	−	+	−	+	−

At each node, the propagated contribution is multiplied by the intrinsic strength of the target argument. This locally moves the resulting justification value towards zero. Its overall effect on a sequence, however, depends on the propagation function: for purely multiplicative options, the attenuation accumulates directly, whereas complement-based options may transform a smaller value into a larger value at the following step. In particular, we note the following.

Support propagation *no longer approaches the upper bound monotonically, because the sequence now reflects both support reinforcement and argument reliability. Although each support reinforces the justification received from the preceding argument, multiplication by the intrinsic strength of the target may reduce the resulting value.*

LC *retains its non-monotonic alternation between lower and higher magnitudes. Intrinsic argument strengths locally attenuate the propagated values, but they do not necessarily smooth every subsequent oscillation, since the LC complementation may turn a smaller magnitude into a larger one at the following step.*

SA *is such that intrinsic argument strengths introduce additional multiplicative factors. Consequently, the magnitude decreases monotonically and more rapidly than when all argument strengths are equal to* 1*.*

CRP *preserves its strong parity-dependent behavior: odd-length attack sequences produce negative values with relatively high magnitudes, whereas even-length sequences produce positive values with relatively low magnitudes. While intrinsic argument strengths mitigate the saturation towards* 1 *at odd steps and towards* 0 *at even steps, the distinction between the two parities remains evident in this example.*

SRP *alternates polarity while its magnitude decreases monotonically towards zero. Since both residual attack factors and intrinsic argument strengths are multiplied along the sequence, their attenuating effects accumulate particularly rapidly.*

CDP *exhibits the parity pattern complementary to that of CRP: odd-length sequences produce negative values with comparatively lower magnitudes, whereas even-length sequences produce positive values with comparatively higher magnitudes. In this example, intrinsic argument strengths substantially moderate the tendency towards the extreme values observed when all argument strengths are equal to* 1*.*

Overall, we can see the effect of intrinsic argument strengths on the overall evaluation. They do not merely introduce a uniform damping effect. Rather, they interact with the specific propagation function: they directly reinforce attenuation in SA and SRP, while in LC, CRP, and CDP their effect is also mediated by the subsequent complement-based transformations.

For SA and SRP, the dialectical effect of a sequence is directly encoded in the sign of the propagated justification value. Indeed, assuming that no value involved in the propagation is zero, we have

$$\operatorname{sign}\big(\mathrm{jvp}(\alpha_{i-1}, \alpha_i)\big) = \operatorname{sign}\big(\mathrm{jv}(\alpha_0)\big) \prod_{i=1}^{k} \operatorname{sign}(p_i).$$

Consequently, when $\mathrm{jv}(\alpha_0) > 0$,

$$\alpha_0 \; g\text{-defends } \alpha_k \iff \mathrm{jvp}(\alpha_{k-1}, \alpha_k) > 0; \qquad \alpha_0 \; g\text{-attacks } \alpha_k \iff \mathrm{jvp}(\alpha_{k-1}, \alpha_k) < 0.$$

If $\mathrm{jv}(\alpha_0) < 0$, the two signs are reversed.

General Case

Finally, let us show the full application of the proposed framework on a general case. A complete account of every evaluation step for every combination of depth and breadth options would require considerable space. We therefore describe only part of the computation in detail, while covering the key steps and the situations relevant to the proposed procedure.

Example 7. *For the argumentation graph in Figure 2, consider the following weights:*

$$\begin{array}{cccccc} (1,2):+0.9 & (2,5):-0.8 & (3,3):-0.5 & (3,5):+0.8 & (4,6):-0.9 & (5,7):-0.8 \\ (5,8):+0.4 & (6,5):+0.7 & (7,9):+0.8 & (7,10):-0.9 & (8,5):-0.7 & (8,6):+0.9 \\ (8,7):-0.8 & (8,13):+0.9 & (9,11):+0.6 & (9,12):-0.7 & (10,8):+0.8 & (10,13):-0.6 \\ (13,14):+0.8 & (14,15):-0.9 & (15,14):+0.7 & & & \end{array}$$

and assume the arguments have the following strengths:

$$\begin{array}{cccccccc} 1{:}0.7 & 2{:}0.6 & 3{:}0.9 & 4{:}0.7 & 5{:}0.5 & 6{:}0.8 & 7{:}0.6 & 8{:}0.8 \\ 9{:}0.9 & 10{:}0.3 & 11{:}1.0 & 12{:}0.4 & 13{:}1.0 & 14{:}0.8 & 15{:}0.7 & \end{array}$$

Let us start by evaluating the interface nodes of cycles in the CC. For each cyclic component, and for each of its interface nodes, we will provide a table describing all the partial computations associated with each node in the sequence obtained by opening the cycle at that interface node, for all jvp *options. Each row in the table will report one step of the partial computation, indicating the type of relationship being processed and the final acceptance value (sign) for the target argument. The* $\downarrow(\alpha,\beta)$ *and* $\xrightarrow{(\alpha,\beta)}$ *symbols are simply visual connectors indicating the flow in the intermediate steps from the former to the latter, followed by the arc* (α,β) *that the step is processing.*

- *The linear justification value of 3 along cycle* $\langle 3 \rangle$ *opened at the interface node 3 is*

	Type	*Acceptance*	*LC*	*SA*	*CRP*	*SRP*	*CDP*
$w_{\mathcal{A}}(3) =$	—	+	*0.9*				
$\mathrm{jvp}((3,3)) =$	*attack*	−	*0.55*	*0.45*	*0.55*	*0.45*	*0.05*

- *Considering cycle* $C_1 = \langle 5,8 \rangle$*, the linear justification value of interface node 5 is*

	Type	*Acceptance*	*LC*	*SA*	*CRP*	*SRP*	*CDP*
$w_{\mathcal{A}}(5) =$	—	+	*0.5*				
$\downarrow(5,8)$	*support*	+	*0.56*				
$\mathrm{jvp}((8,5)) =$	*attack*	−	*0.608*	*0.392*	*0.832*	*0.168*	*0.308*

while for interface node 8 it is

	Type	*Acceptance*	*LC*	*SA*	*CRP*	*SRP*	*CDP*
$w_{\mathcal{A}}(8) =$	—	+	*0.8*				
$\downarrow(8,5)$	*attack*	−	*0.22*	*0.28*	*0.38*	*0.12*	*0.07*
$\mathrm{jvp}((5,8)) =$	*support*	−	*0.532*	*0.568*	*0.628*	*0.472*	*0.442*

- *As regards cycle* $C_2 = \langle 5,8,6 \rangle$*, the linear justification value of interface nodes (5, 8 and 6) is the same for all strategies, because all the arcs in the cycle are supports, and specifically:*

$$w_{\mathcal{A}}(5) = 0.5 \xrightarrow{(5,8)} 0.56 \xrightarrow{(8,6)} 0.7648 \to \mathrm{jvp}((6,5)) = 0.92944$$

$$w_{\mathcal{A}}(8) = 0.8 \xrightarrow{(8,6)} 0.784 \xrightarrow{(6,5)} 0.4676 \rightarrow \text{jvp}((5,8)) = 0.68056$$

$$w_{\mathcal{A}}(6) = 0.8 \xrightarrow{(6,5)} 0.47 \xrightarrow{(5,8)} 0.5456 \rightarrow \text{jvp}((8,6)) = 0.95456$$

For the other cyclic components, we will describe only the computations associated with interface node 8.

- *Consider cycle* $C_3 = \langle 7, 10, 8 \rangle$.

	Type	*Acceptance*	*LC*	*SA*	*CRP*	*SRP*	*CDP*
$w_{\mathcal{A}}(8) =$	—	+			*0.8*		
$\downarrow (8,7)$	*attack*	−	*0.216*	*0.384*	*0.504*	*0.096*	*0.096*
$\downarrow (7,10)$	*attack*	+	*0.24168*	*0.10368*	*0.01448*	*0.00288*	*0.27408*
$\text{jvp}((10,8)) =$	*support*	+	*0.84834*	*0.82074*	*0.80298*	*0.80058*	*0.85482*

- *Cycle* $C_4 = \langle 5, 7, 10, 8 \rangle$.

	Type	*Acceptance*	*LC*	*SA*	*CRP*	*SRP*	*CDP*
$w_{\mathcal{A}}(8) =$	—	+			*0.8*		
$\downarrow (8,5)$	*attack*	−	*0.22*	*0.28*	*0.38*	*0.12*	*0.07*
$\downarrow (5,7)$	*attack*	+	*0.4944*	*0.1344*	*0.0744*	*0.0144*	*0.5664*
$\downarrow (7,10)$	*attack*	−	*0.16651*	*0.03629*	*0.29777*	*0.00043*	*0.11707*
$\text{jvp}((10,8))$	*support*	−	*0.83330*	*0.80726*	*0.85955*	*0.80009*	*0.82341*

- $C_5 = \langle 5, 7, 10, 8, 6 \rangle$.

	Type	*Acceptance*	*LC*	*SA*	*CRP*	*SRP*	*CDP*
$w_{\mathcal{A}}(8) =$	—	+			*0.8*		
$\downarrow (8,6)$	*support*	+			*0.784*		
$\downarrow (6,5)$	*support*	+			*0.4676*		
$\downarrow (5,7)$	*attack*	−	*0.37555*	*0.22445*	*0.54389*	*0.05611*	*0.25555*
$\downarrow (7,10)$	*attack*	+	*0.19860*	*0.06060*	*0.01368*	*0.00168*	*0.23100*
$\text{jvp}((10,8)) =$	*support*	+	*0.83972*	*0.81212*	*0.80274*	*0.80034*	*0.84620*

Let us now move to the acyclic part and the evaluation of the stratified subgraph. Recall that the sequence of layers is $\{1, 3, 4, 7, 8, 10\} \longrightarrow \{2, 6, 9, 13\} \longrightarrow \{5, 11, 12, 14\}$.

- *For Layer 0, nodes 1 and 4 get as a justification value their intrinsic strength:*
  - $jv(1) = w_{\mathcal{A}}(1) = 0.7$*;*
  - $jv(4) = w_{\mathcal{A}}(4) = 0.7$.

  The value for node 3 is obtained by multiplying the jvp *obtained by opening the source cycle* $\langle 3 \rangle$ *in the interface node 3 and the strength of argument 3; from the previous step we have, for the different options:*

	LC	*SA*	*CRP*	*SRP*	*CDP*
$jv(3) =$	*0.495*	*0.405*	*0.495*	*0.405*	*0.045*

For node 8, we must combine, through all options in breadth, the values obtained for each cyclic component that involves 8 (i.e., all five internal cyclic components) by opening it in 8. From the previous step, we have, for the different options in depth (on the columns) and in breadth (on the last rows):

CC	Acceptance	LC	SA	CRP	SRP	CDP
C_1	−	*0.53200*	*0.56800*	*0.62800*	*0.47200*	*0.44200*
C_2	+	*0.68056*	*0.68056*	*0.68056*	*0.68056*	*0.68056*
C_3	+	*0.84834*	*0.82074*	*0.80298*	*0.80058*	*0.85482*
C_4	−	*0.83330*	*0.80726*	*0.85955*	*0.80009*	*0.82341*
C_5	+	*0.83972*	*0.81212*	*0.80274*	*0.80034*	*0.84620*
AJV		+0.16053	+0.15011	+0.12780	+0.16150	+0.17859
MinJV		−0.66664	−0.64581	−0.68764	−0.64007	−0.65873
MaxJV		+0.67867	+0.65659	+0.64238	+0.64046	+0.68386
CJV		+0.05620	+0.05800	+0.03187	+0.07427	+0.07312

Similarly for nodes 7 and 10.

- *Then, the computation moves to layer 1. Node 2 is assigned a justification value by applying the support formula to the justification value of node 1 and arc (1,2); since it has only one incoming arc, all semantics in breadth (*AJV, MinJV, MaxJV, CJV*) coincide:*

  – $jv(2) = (0.7 + 0.3 \cdot 0.9) \cdot 0.6 = 0.97 \cdot 0.6 = 0.582$.

  Node 6 is assigned a justification value by applying jv *(the formula for composition in breadth) to the* jvp*'s associated with all of its incoming arcs, i.e., (4,6) and (8,6). For the former, the attack formula is applied to the current justification value of node 4 and arc (4,6), obtaining:*

Option	Type	Acceptance	LC	SA	CRP	SRP	CDP
$\mathrm{jv}(4) =$	—	+	*0.7*				
$\mathrm{jvp}((4,6)) =$	*attack*	−	*0.37*	*0.63*	*0.93*	*0.07*	*0.27*

For the latter, it is applied to the justification value computed for 8 on the cyclic components and arc $(8,6)$*, which depends on the strategy in breadth, according to the following table:*

AJV	*Type*	*Acceptance*	*LC*	*SA*	*CRP*	*SRP*	*CDP*
$\mathrm{jv}(8) =$	—	+	*0.16053*	*0.15011*	*0.12780*	*0.16150*	*0.17859*
$\mathrm{jvp}((8,6)) =$	*support*	+	*0.91605*	*0.91501*	*0.91278*	*0.91615*	*0.91786*
MaxJV	*Type*	*Acceptance*	*LC*	*SA*	*CRP*	*SRP*	*CDP*
$\mathrm{jv}(8) =$	—	+	*0.67867*	*0.65659*	*0.64238*	*0.64046*	*0.68386*
$\mathrm{jvp}((8,6)) =$	*support*	+	*0.96787*	*0.96566*	*0.96424*	*0.96405*	*0.96839*
MinJV	*Type*	*Acceptance*	*LC*	*SA*	*CRP*	*SRP*	*CDP*
$\mathrm{jv}(8) =$	—	−	*−0.66664*	*−0.64581*	*−0.68764*	*−0.64007*	*−0.65873*
$\mathrm{jvp}((8,6)) =$	*support*	−	*−0.96666*	*−0.96458*	*−0.96876*	*−0.96401*	*−0.96587*
CJV	*Type*	*Acceptance*	*LC*	*SA*	*CRP*	*SRP*	*CDP*
$\mathrm{jv}(8) =$	—	+	*0.05620*	*0.05800*	*0.03187*	*0.07427*	*0.07312*
$\mathrm{jvp}((8,6)) =$	*support*	+	*0.90562*	*0.90580*	*0.90319*	*0.90743*	*0.90731*

So, combining these partial results through jv*, we have the following values for* $\mathrm{jv}(6)$*:*

$\mathrm{jv}(6)$	LC	SA	CRP	SRP	CDP
AJV	+0.21842	+0.11400	−0.00689	+0.33846	+0.25914
MinJV	−0.77333	−0.77166	−0.77501	−0.77121	−0.77270
MaxJV	+0.77429	+0.77253	+0.77139	+0.77124	+0.77471
CJV	+0.42850	+0.22064	−0.02145	+0.66994	+0.50985

Similarly for nodes 9 and 13.

These results illustrate that the choice of the combination functions is semantically consequential, rather than merely producing numerical variations. In particular, the consistently negative values obtained with MinJV and the consistently positive values obtained with MaxJV reflect their respective precedence policies, while the sign changes obtained with some AJV and CJV combinations result from the interaction between their compensatory aggregation and the selected depth-propagation policy.

- *For Layer 2, consider node 5, which is the target of arcs (2,5), (3,5), (6,5) and (8,5) in the graph. For (2,5) and (6,5), the* jvp *is applied to the justification value computed for nodes 2 and 6*

in the previous layer; for (3,5) and (8,5), it is applied to the justification value computed in advance for nodes 3 and 8. Finally, jv *is applied to combine these four values in breadth. The same procedure applies to nodes 11, 12, and 14.*

7. Conclusions

Generalized Argumentation Frameworks were proposed as a unifying Abstract Argumentation Framework capable of combining several extensions of traditional Argumentation Frameworks, most prominently weighted attack/support relationships, while adding further features. Previous work showed how GAFs can simulate several major argumentation frameworks from the literature; however, a specific evaluation strategy designed to exploit the distinctive features of GAFs was still missing. This paper contributes in this direction by proposing an evaluation strategy and several semantics. The core of the strategy consists of a factorization of the argumentation graph, intended to represent the relevant direct and indirect relationships among arguments while avoiding redundant contributions, together with a pair of functions specifying how justification values propagate in depth and are combined in breadth. Different definitions of these functions determine different semantics; this paper investigates several options. The proposed approach has several features that may be useful in research on argumentation frameworks, including the following:

- It exploits the additional expressive features of GAFs;
- It provides both a binary and a graded outcome (with a justification value associated with the position to which the argument belongs), combining aspects of extension-based, ranking-based, and labeling-based semantics;
- It returns a single extension for each choice of the functions (i.e., for each semantics), thereby avoiding the choice indeterminacy that may arise when multiple, potentially incomparable extensions are returned;
- It separates the justification status of the arguments from their justification value;
- The argumentation framework is evaluated progressively, allowing intermediate results to be reused.

The behavior of the approach has been analyzed theoretically and illustrated through sample cases. These analyses highlight several properties of the proposed semantics and show how their behavior relates to the intended interpretations.

Existing argumentation frameworks have been applied to real-world problems, including the analysis of judicial cases, the reconstruction of reading order in documents, and the comparison of alternative analogical mappings. The additional features of GAFs may provide further modelling possibilities in such settings. Future work will therefore include applying the framework to both artificial and real-world cases.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

Acknowledgments: During the preparation of this manuscript, the author used Work GPT-5.6 Sol Light for the purposes of a general final check and fixing of typos and grammar. The authors have reviewed and edited the output and take full responsibility for the content of this publication.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AF	Argumentation Framework
BAF	Bipolar Argumentation Framework
BWAF	Bipolar Weighted Argumentation Framework
CC	Cycle Cover
GAF	Generalized Argumentation Framework
WAF	Weighted Argumentation Framework

References

1. Toulmin, S.E. *The Uses of Argument*; Cambridge University Press: Cambridge, UK, 1958.
2. Dung, P.M. On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and N-Person Games. *Artif. Intell.* **1995**, *77*, 321–357. [CrossRef]
3. Ferilli, S. Introducing General Argumentation Frameworks and Their Use. In *Proceedings of the AIxIA 2020–Advances in Artificial Intelligence, Virtual, 25–27 November 2020*; Lecture Notes in Artificial Intelligence; Springer: Cham, Switzerland, 2020; Volume 12414, pp. 136–153.
4. Cayrol, C.; Lagasquie-Schiex, M. On the acceptability of arguments in bipolar argumentation frameworks. In Proceedings of the ECSQARU, Barcelona, Spain, 6–8 July 2005; Volume 3571, pp. 378–389.
5. Dunne, P.E.; Hunter, A.; McBurney, P.; Parsons, S.; Wooldridge, M. Weighted Argument Systems: Basic Definitions, Algorithms, and Complexity Results. *Artif. Intell.* **2011**, *175*, 457–486. [CrossRef]
6. Amgoud, L.; en Naim, J. Weighted bipolar argumentation graphs: Axioms and semantics. In Proceedings of the IJCAI'18: Proceedings of the 27th International Joint Conference on Artificial Intelligence, Stockholm, Sweden, 13–19 July 2018.
7. Pazienza, A.; Ferilli, S.; Esposito, F. Constructing and Evaluating Bipolar Weighted Argumentation Frameworks for Online Debating Systems. In Proceedings of the 1st Workshop on Advances in Argumentation in Artificial Intelligence, AI^3 2017 @ AI*IA 2017, Bari, Italy, 16–17 November 2017; pp. 111–125.
8. Pazienza, A.; Ferilli, S.; Esposito, F. On the Gradual Acceptability of Arguments in Bipolar Weighted Argumentation Frameworks with Degrees of Trust. In Proceedings of the Foundations of Intelligent Systems—23rd International Symposium, ISMIS, Warsaw, Poland, 26–29 June 2017; pp. 195–204.
9. Amgoud, L.; Ben-Naim, J. Axiomatic foundations of acceptability semantics. In Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning, KR, Cape Town, South Africa, 25–29 April 2016; Volume 16.
10. Amgoud, L.; Cayrol, C. A reasoning model based on the production of acceptable arguments. *Ann. Math. Artif. Intell.* **2002**, *34*, 197–215. [CrossRef]
11. Bench-Capon, T.J.M. Persuasion in practical argument using value-based argumentation frameworks. *J. Log. Comput.* **2003**, *13*, 429–448. [CrossRef]
12. Parsons, S.; Tang, Y.; Sklar, E.; McBurney, P.; Cai, K. Argumentation-based reasoning in agents with varying degrees of trust. In *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2, Taipei, Taiwan, 2–6 May 2011*; International Foundation for Autonomous Agents and Multiagent Systems: Richland County, SC, USA, 2011; Volume 2, pp. 879–886.
13. Ferilli, S. An Attack/Support-aware Weight Composition Strategy for Bipolar Weighted Argumentation Frameworks. In *Foundations of Intelligent Systems, Proceedings of ISMIS 2026*; Lecture Notes in Artificial Intelligence; Springer International Publishing: Cham, Switzerland, 2026; Volume 16685, pp. 24–34.
14. Baroni, P.; Caminada, M.; Giacomin, M. An introduction to argumentation semantics. *Knowl. Eng. Rev.* **2011**, *26*, 365–410. [CrossRef]
15. Caminada, M.; Gabbay, D.M. A logical account of formal argumentation. *Stud. Log.* **2009**, *93*, 109–145. [CrossRef]
16. Bonzon, E.; Delobelle, J.; Konieczny, S.; Maudet, N. A Comparative Study of Ranking-Based Semantics for Abstract Argumentation. In Proceedings of the AAAI, Phoenix, AZ, USA, 12–17 February 2016; pp. 914–920.
17. Amgoud, L.; Ben-Naim, J. Ranking-based semantics for argumentation frameworks. In *Proceedings of the International Conference on Scalable Uncertainty Management (SUM)*; Lecture Notes in Artificial Intelligence; Springer: Berlin/Heidelberg, Germany, 2013; Volume 8078, pp. 134–147.
18. Bistarelli, S.; Taticchi, C. A Labelling Semantics for Weighted Argumentation Frameworks. In Proceedings of the 35th Italian Conference on Computational Logic-CILC 2020, Rende, Italy, 13–15 October 2020; pp. 263–277.
19. Berge, C. *Graphs and Hypergraphs*; Elsevier Science Ltd.: Oxford, UK, 1985.
20. Johnson, D.B. Finding All the Elementary Circuits of a Directed Graph. *SIAM J. Comput.* **1975**, *4*, 77–84. [CrossRef]
21. Eades, P.; Sugiyama, K. How to draw a directed graph. *J. Inf. Process.* **1990**, *13*, 424–437.

22. Kahn, A.B. Topological sorting of large networks. *Commun. ACM* **1962**, *5*, 558–562. [CrossRef]
23. Pazienza, A.; Ferilli, S. The Linear Algebra of Abstract Argumentation. In *Proceedings of the 2nd Workshop on Advances in Argumentation in Artificial Intelligence, Co-Located with XVII International Conference of the Italian Association for Artificial Intelligence, AI^3@AI*IA 2018, Trento, Italy, 20–23 November 2018*; CEUR Workshop Proceedings; CEUR-WS.org: Bonn, Germany, 2018; Volume 2296, pp. 71–85.
24. Xu, Y.; Cayrol, C. The matrix approach for abstract argumentation frameworks. In *Proceedings of the TAFA @ IJCAI*; Springer: Cham, Switzerland, 2015; pp. 243–259.
25. Bistarelli, S.; Tappini, A.; Taticchi, C. A Matrix Approach for Weighted Argumentation Frameworks. In Proceedings of the FLAIRS 2018, Melbourne Beach, FL, USA, 21–23 May 2018; pp. 507–512.
26. Hadjisoteriou, E. Computing Argumentation with Matrices. In *Proceedings of the ICCSW 2015, London, UK, 24–25 September 2015*; Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik: Wadern, Germany, 2015; Volume 49, pp. 29–36.
27. Pu, F.; Luo, G.; Chen, Y. Boolean Matrix Approach for Abstract Argumentation. In *Multi-Agent Systems and Agreement Technologies*; Springer: Cham, Switzerland, 2017; pp. 470–480.
28. Corea, C.; Thimm, M. Using Matrix Exponentials for Abstract Argumentation. In Proceedings of the SAFA @ COMMA, Potsdam, Germany, 13 September 2016; pp. 10–21.
29. Baroni, P.; Rago, A.; Toni, F. How Many Properties Do We Need for Gradual Argumentation? In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*; AAAI Press: Palo Alto, CA, USA, 2018; pp. 1736–1743.
30. Potyka, N. Extending Modular Semantics for Bipolar Weighted Argumentation. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2019), Montreal, QC, Canada, 13–17 May 2019*; International Foundation for Autonomous Agents and Multiagent Systems: Richland, SC, USA, 2019; pp. 1722–1730.
31. Thieyre, J.; Anaissy, C.A.; Beynier, A.; Destercke, S.; Maudet, N.; Vesic, S. Uncertainty in Quantitative Bipolar Argumentation Frameworks. In *Frontiers in Artificial Intelligence and Applications, Proceedings of the 28th European Conference on Artificial Intelligence (ECAI 2025)-Including 14th Conference on Prestigious Applications of Intelligent Systems (PAIS 2025)*; IOS Press: Amsterdam, The Netherlands, 2025; Volume 413, pp. 1679–1686.
32. Wang, Z.; Shen, Y. Convergent Semantics for Weighted Bipolar Argumentation. In Proceedings of the Fortieth AAAI Conference on Artificial Intelligence (AAAI-26), Singapore, 20–27 January 2026; Volume 40, pp. 19415–19423. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.