Logical Encoding of Argumentation Frameworks with Higher-order Attacks and Evidential Supports: Taking into account the Collective Interactions

Marie-Christine Lagasquie-Schiex

Logical Encoding of Argumentation Frameworks with Higher-order Attacks and Evidential Supports: Taking into account the Collective Interactions

M-Christine Lagasquie-Schiex

Université de Toulouse, IRIT,
118 route de Narbonne, 31062 Toulouse, France
{lagasq}@irit.fr

Tech. Report
IRIT/RR- -2021- -05- -FR

Juin 2021

Abstract

In [11], we have proposed a logical encoding of argumentation frameworks with higher-order interactions (*i.e.* attacks/supports whose targets are arguments or other attacks/supports) with an evidential meaning for supports (such frameworks are called REBAF). Then this encoding has been used for giving a characterization of REBAF semantics. In this work, two constraints have been given: first the source of an interaction must be a single argument, second no support cycle is authorized in the REBAF.
In [20], a first improvement of this work has been done in order to relax the second constraint, correcting the first attempt done in [11] and allowing to take into account REBAF with any kind of support cycles.

In this new work, we propose a second improvement of our previous works in order to relax the first constraint and so to be able to take into account collective interactions, *i.e.* those whose source can be a set of arguments and not only one single argument.

Contents

1 Introduction

Formal argumentation has become an essential paradigm in Artificial Intelligence, *e.g.* for reasoning from incomplete and/or contradictory information or for modelling the interactions between agents [26]. Formal abstract frameworks have greatly eased the modelling and study of argumentation. The original Dung's argumentation framework (AF) [16] consists of a collection of *arguments* interacting with each other through a relation reflecting conflicts between them, called *attack*, and enables to determine *acceptable* sets of arguments called *extensions*.

AF have been extended along different lines, *e.g.* by enriching them with positive interactions between arguments (usually expressed by a support relation), or higher-order interactions (*i.e.* interactions whose targets are other interactions).

Positive interactions between arguments. They have been first introduced in [19, 27]. In [9], the support relation is left general so that the bipolar framework keeps a high level of abstraction. The associated semantics are based on the combination of the attack relation with the support relation which results in new complex attack relations. However, there is no single interpretation of the support, and a number of researchers proposed specialized variants of the support relation (deductive support [4], necessary support [22, 23], evidential support [24, 25]). Each specialization can be associated with an appropriate modelling using an appropriate complex attack. These proposals have been developed quite independently, based on different intuitions and with different formalizations. [10] presents a comparative study in order to restate these proposals in a common setting, the bipolar argumentation framework (see also [12] for another survey).

Higher-order interactions. The idea of encompassing attacks to attacks in abstract argumentation frameworks has been first considered in [3] in the context of an extended framework handling argument strengths and their propagation. Then, higher-order attacks have been considered for representing preferences between arguments (second-order attacks in [21]), or for modelling situations where an attack might be defeated by an argument, without contesting the acceptability of the source of the attack [2]. Attacks to attacks and supports have been first considered in [17] with higher level networks, then in [28]; and more generally, [13] proposes an Attack-Support Argumentation Framework which allows for nested attacks and supports, *i.e.* attacks and supports whose targets can be other attacks or supports, at any level.

Here are examples of higher-order interactions in the legal field. The first example considers only higher-order attacks (this example is borrowed from [1]).

Example 1 *The lawyer says that the defendant did not have intention to kill the victim (argument b). The prosecutor says that the defendant threw a sharp knife towards the victim (argument a). So, there is an attack from a to b. And the intention to kill should be inferred. Then the lawyer says that the defendant was in a habit of throwing the knife at his wife's foot once drunk. This latter argument (argument c) is better considered attacking the attack from a to b, than argument a itself. Now the prosecutor's argumentation seems no longer sufficient for proving the intention to kill.* □

The second example is a variant of the first one and considers higher-order attacks and evidential supports.

Example 2 *The prosecutor says that the defendant had intention to kill the victim (argument b). A witness says that she saw the defendant throwing a sharp knife towards the victim (argument a). Argument a can be considered as a support for argument b. The lawyer argues back that the defendant was in a habit of throwing the knife at his wife's foot once drunk. This latter argument (argument c) is better considered attacking the support from a to b, than arguments a or b themselves. Once again, the prosecutor's argumentation seems no longer sufficient for proving the intention to kill.* □

We follow here an evidential understanding of the support relation [24] that allows to distinguish between two different kinds of arguments: *prima-facie* and *standard arguments*. *Prima-facie* arguments were already present in [27] as those that are justified whenever they are not defeated. On the other hand, *standard arguments* are not directly assumed to be justified and must inherit support from prima-facie arguments through a "chain" of supports. For instance, in Example 2, arguments a and c could be considered as prima-facie arguments while b would be regarded

as a standard argument. Hence, while a and c can be accepted as in Dung's argumentation, b must inherit support from a: this holds if c is not accepted, but does not otherwise. Indeed, in the latter, the support from a to b is defeated by c.

A natural idea that has proven useful to define semantics for these extended frameworks, known as "flattening technique", consists in turning the original extended framework into an AF, by introducing meta-arguments and a new simple (first-order) attack relation involving these meta-arguments [2, 5, 6, 13], or by reducing higher-order attacks to first-order joint attacks [18]. More recently, alternative acceptability semantics have been defined in a direct way for argumentation frameworks with higher-order attacks [7] or for higher-order attacks and supports (necessary supports: [14], evidential supports: [8]). The idea is to specify the conditions under which the arguments (resp. the interactions) are considered as accepted directly on the extended framework, without translating the original framework into an AF.

Morever, in [11] then in [20], a logical encoding of argumentation frameworks with higher-order attacks and evidential supports (REBAF) has been proposed. This encoding is able to take into account REBAF. A strong constraint exists in these works: the source of an interaction must be a single argument. So the aim of the current work is to relax this constraint and to propose a new logical encoding of REBAF in which interactions are not binary ones (this kind of interactions is called "collective interactions").

The paper is organized as follows: the necessary background about argumentation frameworks is given in Section 2; the logical encoding for frameworks with higher-order attacks and evidential supports (REBAF) is recalled in Section 3; the new proposition that can handle collective interactions is given in Section 4; Section 5 concludes the paper. The proofs are given in Appendix A.

2 Background on argumentation frameworks

Note that the text (definitions, propositions and examples) of this section is extracted from [11].

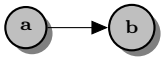
2.1 The Standard Abstract Framework

The standard case handles only one kind of interaction: attacks between arguments.

Definition 1 **[15]** *A* Dung's argumentation framework (AF) *is a tuple* $\mathbf{AF} = \langle \mathbf{A}, \mathbf{R} \rangle$*, where* $\mathbf{A}$ *is a finite and non-empty set of* arguments *and* $\mathbf{R} \subseteq \mathbf{A} \times \mathbf{A}$ *is a binary* attack relation *on the arguments, with* $(a, b) \in \mathbf{R}$ *indicates that* a *attacks* b*.*

A graphical representation can be used for an AF.

Example 3 *An attack* $(a, b) \in \mathbf{R}$ *is represented by two nodes* a*,* b *(in a circle) and a simple edge from* a *to* b*:*



□

We recall the definitions[1] of some well-known extension-based semantics. Such a semantics specifies the requirements that a set of arguments should satisfy. The basic requirements are the following ones:

- An extension can "stand together". This corresponds to the conflict-freeness principle.
- An extension can "stand on its own", namely is able to counter all the attacks it receives. This corresponds to the defence principle.
- Reinstatement is a kind of dual principle. An attacked argument which is defended by an extension is reinstated by the extension and should belong to it.
- Stability: an argument that does not belong to an extension must be attacked by this extension.

[1] Where "iff" (resp. "w.r.t.") stands for "if and only if" (resp. "with respect to").

Definition 2 **[15]** *Let* $\mathbf{AF} = \langle \mathbf{A}, \mathbf{R} \rangle$ *and* $S \subseteq \mathbf{A}$.

- S *is* conflict-free *iff* $(a, b) \notin \mathbf{R}$ *for all* $a, b \in S$.
- $a \in \mathbf{A}$ *is* acceptable *w.r.t.* S *(or equivalently* S defends a*) iff for each* $b \in \mathbf{A}$ *with* $(b, a) \in \mathbf{R}$*, there is* $c \in S$ *with* $(c, b) \in \mathbf{R}$.
- *The* characteristic function $\mathcal{F}$ *of* **AF** *is defined by:* $\mathcal{F}(S) = \{a \in \mathbf{A}$ *such that* a *is acceptable w.r.t.* $S\}$.
- S *is* admissible *iff* S *is conflict-free and* $S \subseteq \mathcal{F}(S)$.
- S *is* a complete extension *of* **AF** *iff it is conflict-free and a fixed point of* $\mathcal{F}$.
- S *is* the grounded extension *of* **AF** *iff it is the minimal (w.r.t.* $\subseteq$*) fixed point*[2] *of* $\mathcal{F}$.
- S *is* a preferred extension *of* **AF** *iff it is a maximal (w.r.t.* $\subseteq$*) complete extension.*
- S *is* a stable extension *of* **AF** *iff it is conflict-free and for each* $a \notin S$*, there is* $b \in S$ *with* $(b, a) \in \mathbf{R}$.

Note that the complete (resp. grounded, preferred, stable) semantics satisfies the conflict-freeness, defence and reinstatement principles.

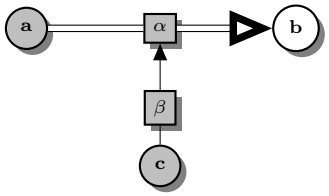
2.2 A Framework with Higher-Order Evidential Supports and Attacks

In this section, we recall the extension of [7] proposed in [8] for handling recursive attacks and evidence-based supports.

Definition 3 **[8]** *An* evidence-based recursive argumentation framework (REBAF) *is a sextuple* $\langle \mathbf{A}, \mathbf{R}_a, \mathbf{R}_e, \mathbf{s}, \mathbf{t}, \mathbf{P} \rangle$ *where* $\mathbf{A}$*,* $\mathbf{R}_a$ *and* $\mathbf{R}_e$ *are three (possible infinite) pairwise disjunct sets respectively representing arguments, attacks and supports names, and where* $\mathbf{P} \subseteq \mathbf{A} \cup \mathbf{R}_a \cup \mathbf{R}_e$ *is a set representing the* prima-facie *elements that do not need to be supported. Functions* $\mathbf{s} : (\mathbf{R}_a \cup \mathbf{R}_e) \longrightarrow 2^{\mathbf{A}} \setminus \varnothing$ *and* $\mathbf{t} : (\mathbf{R}_a \cup \mathbf{R}_e) \longrightarrow (\mathbf{A} \cup \mathbf{R}_a \cup \mathbf{R}_e)$ *respectively map each attack and support to its source and its target.*

Note that the source of attacks and supports is a set of arguments, the set $\mathbf{P}$ may contain several prima-facie elements (arguments, attacks and supports) and no constraint on the prima-facie elements is assumed (they can be attacked or supported).

Example 2 (cont'd): The argumentation framework corresponding to the second example given in the introduction can be represented as follows (argument names are given in circular nodes, interaction names in square nodes, prima-facie elements are in grey nodes and non prima-facie element in white nodes; supports are represented by double edges):



□

Semantics of REBAF are defined in [8] using the extension of the notion of structure introduced in [7]. The idea is to characterize which arguments are regarded as "acceptable", and which attacks and supports are regarded as "valid", with respect to some structure.

Consider a given framework $\mathbf{REBAF} = \langle \mathbf{A}, \mathbf{R}_a, \mathbf{R}_e, \mathbf{s}, \mathbf{t}, \mathbf{P} \rangle$.

[2] It can be proved that the minimal fixed point of $\mathcal{F}$ is conflict-free.

Definition 4 **[8]** *A triple $U = (S, \Gamma, \Delta)$ is said to be a* structure *of* **REBAF** *iff it satisfies: $S \subseteq \mathbf{A}$, $\Gamma \subseteq \mathbf{R}_a$ and $\Delta \subseteq \mathbf{R}_e$.*

Intuitively, the set S represents the set of "acceptable" arguments w.r.t. the structure U, while Γ and Δ respectively represent the set of "valid attacks" and "valid supports" w.r.t. U. Any attack[3] $\alpha \in \overline{\Gamma}$ is understood as "non-valid" and, in this sense, it cannot defeat the element that it is targeting. Similarly, any support $\beta \in \overline{\Delta}$ is understood as "non-valid" and it cannot support the element that it is targeting.

The following definitions are extensions of the corresponding ones defined in [7] in order to take into account the evidential supports.

Definition 5 **[8]** *Given a structure $U = (S, \Gamma, \Delta)$,*

- *The sets of defeated elements w.r.t. U are:*

$$Def_X(U) \stackrel{\text{def}}{=} \{x \in X | \exists \alpha \in \Gamma, \mathbf{s}(\alpha) \subseteq S \text{ and } \mathbf{t}(\alpha) = x\}$$

with $X \in \{\mathbf{A}, \mathbf{R}_a, \mathbf{R}_e\}$

$$Def(U) \stackrel{\text{def}}{=} Def_{\mathbf{A}}(U) \cup Def_{\mathbf{R}_a}(U) \cup Def_{\mathbf{R}_e}(U)$$

- *The set of supported elements $Sup(U)$ is recursively defined as follows:*[4]

$$Sup(U) \stackrel{\text{def}}{=} \mathbf{P} \cup \{\mathbf{t}(\alpha) | \exists \alpha \in \Delta \cap Sup(U \backslash \{\mathbf{t}(\alpha)\}), \mathbf{s}(\alpha) \subseteq (S \cap Sup(U \backslash \{\mathbf{t}(\alpha)\}))\}$$

Note that a standard element is supported if there is a "chain"[5] of supported supports leading to it, rooted in prima-facie arguments. Acceptability is more complex. Intuitively, an element is *acceptable* if it supported and in addition, every attack against it can be considered as "non-valid" because either the source or the attack itself is defeated or cannot be supported.

The elements that cannot be supported w.r.t. a structure U are called *unsupportable* w.r.t. U. An element is *supportable* w.r.t. U if there is a support for it which is non-defeated by U, with its source being non-defeated by U, and the support and its source being in turn supportable.

The elements that are defeated or unsupportable are called *unacceptable*.

Then an attack is said *unactivable* if either some argument in its source or itself is unacceptable.

Formally,

- The set of *unsupportable* elements w.r.t. U is:

$$UnSupp(U) \stackrel{\text{def}}{=} \overline{Sup(U')}$$

with $U' = (\overline{Def_{\mathbf{A}}(U)}, \mathbf{R}_a, \overline{Def_{\mathbf{R}_e}(U)})$.

- The set of *unacceptable* elements w.r.t. U is:

$$UnAcc(U) \stackrel{\text{def}}{=} Def(U) \cup UnSupp(U)$$

- The set of *unactivable* attacks w.r.t. U is:

$$UnAct(U) \stackrel{\text{def}}{=} \{\alpha \in \mathbf{R}_a | \alpha \in UnAcc(U) \text{ or } \mathbf{s}(\alpha) \cap UnAcc(U) \neq \varnothing\}$$

Definition 6 **[8]** *An element $x \in \mathbf{A} \cup \mathbf{R}_a \cup \mathbf{R}_e$ is said to be* acceptable *w.r.t. a structure U iff (i) $x \in Sup(U)$ and (ii) every attack $\alpha \in \mathbf{R}_a$ with $\mathbf{t}(\alpha) = x$ is unactivable, that is, $\alpha \in UnAct(U)$.*

[3] By $\overline{\Gamma} \stackrel{\text{def}}{=} \mathbf{R}_a \backslash \Gamma$ we denote the set complement of Γ w.r.t. $\mathbf{R}_a$. Similarly, by $\overline{\Delta} \stackrel{\text{def}}{=} \mathbf{R}_e \backslash \Delta$ we denote the set complement of Δ w.r.t. $\mathbf{R}_e$.

[4] By abuse of notation, we write $U \backslash T$ instead of $(S \backslash T, \Gamma \backslash T, \Delta \backslash T)$ with $T \subseteq (\mathbf{A} \cup \mathbf{R}_a \cup \mathbf{R}_e)$.

[5] In fact, strictly speaking, this chain is in reality a "tree". That is due to several reasons. The first one is that each support can be in turn supported. And the second reason is the fact that interactions are collective; so the source of a support can be a set of arguments and in this case all elements in the source are needed for supporting the target.

$Acc(U)$ denotes the set containing all arguments, attacks and supports that are acceptable with respect to U.

The following order relations will help defining preferred structures: for any pair of structures $U = (S, \Gamma, \Delta)$ and $U' = (S', \Gamma', \Delta')$, we write $U \subseteq U'$ iff $(S \cup \Gamma \cup \Delta) \subseteq (S' \cup \Gamma' \cup \Delta')$. As usual, we say that a structure U is $\subseteq$-maximal (resp. $\subseteq$-minimal) iff every U' that satisfies $U \subseteq U'$ (resp. $U' \subseteq U$) also satisfies $U' \subseteq U$ (resp. $U \subseteq U'$).

Definition 7 **[8]** *A structure $U = (S, \Gamma, \Delta)$ is:*

1. self-supporting *iff* $(S \cup \Gamma \cup \Delta) \subseteq Sup(U)$,
2. conflict-free *iff* $X \cap Def_Y(U) = \varnothing$ *for any* $(X, Y) \in \{(S, \mathbf{A}), (\Gamma, \mathbf{R}_a), (\Delta, \mathbf{R}_e)\}$,
3. admissible *iff it is conflict-free and* $S \cup \Gamma \cup \Delta \subseteq Acc(U)$,
4. complete *iff it is conflict-free and* $Acc(U) = S \cup \Gamma \cup \Delta$,
5. grounded *iff it is a $\subseteq$-minimal complete structure,*[6]
6. preferred *iff it is a $\subseteq$-maximal admissible structure,*
7. stable[7] *iff* $(S \cup \Gamma \cup \Delta) = \overline{UnAcc(U)}$.

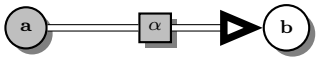
From the above definitions, it follows that if U is a conflict-free structure, unsupportable elements w.r.t. U are not supported w.r.t. U, that is $UnSupp(U) \subseteq \overline{Sup(U)}$.

Note that every admissible structure is also self-supporting. Moreover, the usual relations between extensions also hold for structures: every complete structure is also admissible, every preferred structure is also complete, and every stable structure is also preferred and so admissible. Other properties of REBAF are described in [8], which enable to prove for instance that there is a unique grounded structure.

The previous definitions are illustrated on the following examples.

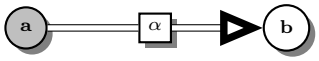
Example 4 *Consider two arguments a and b and a support from a to b. Following the set of prima-facie elements, different behaviours can be described.*

1. *The support and its source are assumed to be prima-facie. The target is not prima-facie.*



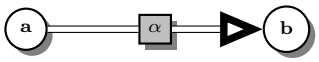
In this case, as α (resp. a) is prima-facie and not attacked, it is acceptable w.r.t. any structure. In contrast, b is not prime-facie, so b is supported w.r.t. a structure U implies that U contains the support α and its source a. As a consequence, the structures $(\{a\}, \varnothing, \{\alpha\})$ and $(\{a, b\}, \varnothing, \{\alpha\})$ are admissible, whereas the structure $(\{b\}, \varnothing, \{\alpha\})$ is not admissible.

2. *Only the source of the support is assumed to be prima-facie.*



In this case, for any structure U, α is not supported w.r.t. U. It is the same for b. So the only admissible structures are $U = (\varnothing, \varnothing, \varnothing)$ and $U = (\{a\}, \varnothing, \varnothing)$.

3. *Only the support is assumed to be prima-facie.*

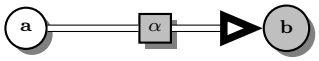


In this case, α is acceptable w.r.t. any structure. However, for any structure U, a is not supported w.r.t. U. So b cannot be supported. As a consequence, the only admissible structures are $U = (\varnothing, \varnothing, \varnothing)$ and $U = (\varnothing, \varnothing, \{\alpha\})$.

[6] The definition for the grounded extension is not given in [8] but can be easily proposed following the definition used in the AF case.

[7] Note that this is also equivalent to U is self-supporting, conflict-free and $\overline{S \cup \Gamma \cup \Delta} \subseteq UnAcc(U)$.

4. *The support and its target are assumed to be prima-facie. The source is not prima-facie.*



In this case, α (resp. b) is acceptable w.r.t. any structure. In contrast, a cannot be supported. So there are 4 admissible structures: $U = (\varnothing, \varnothing, \varnothing)$, $U = (\varnothing, \varnothing, \{\alpha\})$, $U = (\{b\}, \varnothing, \varnothing)$ and $U = (\{b\}, \varnothing, \{\alpha\})$.

□

In the next example, the support is itself the target of an attack.

Example 2 (cont'd): In this framework, neither β nor its source is attacked and β and its source are prima-facie. So, for any structure U, it holds that neither β nor its source c is unacceptable w.r.t. U. As a consequence, for any structure U, α is not acceptable w.r.t. U as α is attacked by β and β is not unactivable w.r.t. U.
As b is not prima-facie, and α is the only support to b, no admissible structure contains b. As a consequence, there is a unique complete, preferred and stable structure $U = (\{a, c\}, \{\beta\}, \varnothing)$. □

Finally, REBAF is a conservative generalization of RAF described in [7] with the addition of supports and joint attacks. Every RAF can be easily translated into a corresponding REBAF with no support and where every element (argument or attack) is prima-facie (see [8]).

In [20], some notions related to directed cycles of supports have been defined in order to take into account support cycles in the logical computation of structures for the REBAF. Nevertheless these definitions are given only for REBAF without collective interactions (so the source of an interaction is only a singleton).

Definition 8 *Let* REBAF $= \langle \mathbf{A}, \mathbf{R}_a, \mathbf{R}_e, \mathbf{s}, \mathbf{t}, \mathbf{P} \rangle$. *A* directed cycle of supports (DCS) *in this REBAF is a sequence* $\mathbf{C} = (x_0, \ldots, x_{n-1}, x_n)$ *such that:*[8]

- $n > 0$ *and* n *is the* size of the DCS*,*
- $\forall i = 0 \ldots n$*,* $x_i \in \mathbf{A} \cup \mathbf{R}_e$*,*
- $x_n = x_0$
- $\forall i = 0 \ldots n-1$*, if* $x_i \in \mathbf{A}$ *then* $x_{i+1} \in \mathbf{R}_e$ *and* $\mathbf{s}(x_{i+1}) = x_i$*,*
- $\forall i = 0 \ldots n-1$*, if* $x_i \in \mathbf{R}_e$ *then* $x_{i+1} = \mathbf{t}(x_i)$*.*

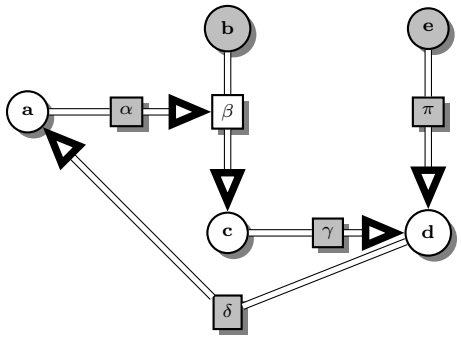
A simple DCS $\mathbf{C} = (x_0, \ldots, x_{n-1}, x_n)$ *is a DCS in which* $\forall i, j = 0 \ldots n-1$*, if* $i \neq j$ *then* $x_i \neq x_j$*.*
An input support of a DCS $\mathbf{C} = (x_0, \ldots, x_{n-1}, x_n)$ *is:*

- *either a support* $y \in \mathbf{R}_e$ *such that* $y \notin \mathbf{C}$ *and* $\exists x_i \in \mathbf{C}$ *and* $x_i = \mathbf{t}(y)$*,*
- *or an argument* $y \in \mathbf{A}$ *such that* $y \notin \mathbf{C}$ *and* $\exists x_i \in \mathbf{R}_e \cap \mathbf{C}$ *and* $y = \mathbf{s}(x_i)$*.*

The set of inputs of the DCS $\mathbf{C}$ *is denoted by* $\mathbf{C}^{In}$ *and it is partitioned into* $\mathbf{C}^{In}_{\mathbf{A}} = \mathbf{C}^{In} \cap \mathbf{A}$ *and* $\mathbf{C}^{In}_{\mathbf{R}_e} = \mathbf{C}^{In} \cap \mathbf{R}_e$*.*

Example 5 *This example illustrates the fact that a support in a cycle can also be the target of another support in the cycle. Note that the source of the targeted support does not belong to the cycle. Here there exists one DCS* $\mathbf{C} = (a, \alpha, \beta, c, \gamma, d, \delta, a)$ *with* $\mathbf{C}^{In} = \{b, \pi\}$*. Note that the source of* π *is not considered as an input of the cycle.*

[8] By abuse of language, the set of the elements composing $\mathbf{C}$ will be also denoted by $\mathbf{C}$. So $\mathbf{C}$ will be used with set operators as $\cap$ ou $\cup$ and will be comparable with other sets.



Another notion will be important in order to compute the logical characterization of REBAF semantics: the *impacting support chains* for an element of a REBAF. Unformally an impacting support chain for an element x is a sequence targeting x, originated in a prima-facie argument and composed alternatively by "an argument, a support, an argument, a support, ...". Moreover no repetition is authorized (so any element appears only one time in the sequence); and x cannot belong to the sequence. So Formally we have:

Definition 9 *Let* $\text{REBAF} = \langle \mathbf{A}, \mathbf{R}_a, \mathbf{R}_e, \mathbf{s}, \mathbf{t}, \mathbf{P} \rangle$. *Let* x *be an element of this REBAF. An* impacting support chain *for* x *is a sequence* $\mathbf{ISC} = (x_0, \ldots, x_n)$ *with* $n > 0$ *and:*

- $\forall x_i, i \in [0 \ldots n]$, $x_i \in (\mathbf{A} \cup \mathbf{R}_e) \setminus \{x\}$
- $x_0 \in \mathbf{A} \cap \mathbf{P}$ *and* $x_n \in \mathbf{R}_e$ *such that* $\mathbf{t}(x_n) = x$
- $\forall i, j \in [0 \ldots n]$, *if* $i \neq j$, *then* $x_i \neq x_j$
- $\forall i \in [1 \ldots n]$, *if* $x_i \in \mathbf{R}_e$ *then* $x_{i-1} = \mathbf{s}(x_i)$
- $\forall i \in [2 \ldots n-1]$, *if* $x_i \in \mathbf{A}$ *then* $x_i = \mathbf{t}(x_{i-1})$

It is obvious to see that if a DCS has some inputs then these inputs may belong to some impacting support chains of the elements of the DCS, if they are prima-facie or if they have at least one impacting support chain.

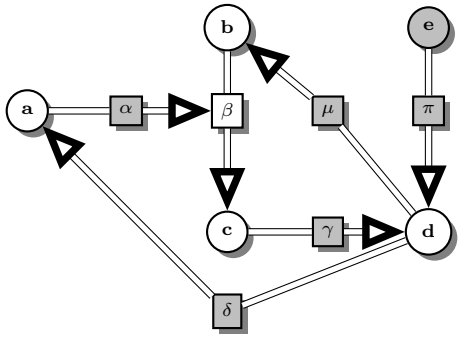
Another trivial property is the fact that, in a DCS without input and in which none argument is prima-facie, the elements of the DCS have no impacting support chains.

Example 5 (cont'd): Considering the impacting support chains of some elements of the DCS, we have for instance:

- For argument d, there exist two impacting support chains: (e, π) and (b, β, c, γ).
- For argument c, there exists only one impacting support chain: (b, β).
- For argument a, there exist two impacting support chains: (e, π, d, δ) and $(b, \beta, c, \gamma, d, \delta)$.
- For support β, there exists only one impacting support chain: $(e, \pi, d, \delta, a, \alpha)$.

□

Example 6 *This example extends Example 5 by adding a second cycle including the source of the support targeted in the first DCS.*



$\mathbf{C} = (a, \alpha, \beta, c, \gamma, d, \delta, a)$ *with* $\mathbf{C}^{In} = \{b, \pi\}$ *and* $\mathbf{C}' = (b, \beta, c, \gamma, d, \mu, b)$ *with* $\mathbf{C}'^{In} = \{\alpha, \pi\}$ *are the two simple DCS.*

Considering the impacting support chains of some elements of the DCS, we have for instance:

- *For argument* d*, there exists only one impacting support chain:* (e, π).
- *For argument* c*, there exists only one impacting support chain:* $(e, \pi, d, \mu, b, \beta)$.
- *For argument* a*, there exists only one impacting support chain:* (e, π, d, δ).
- *For support* β*, there exists only one impacting support chain:* $(e, \pi, d, \delta, a, \alpha)$.

It is worth to notice that DCS can be aggregated:

Definition 10 *Let* $\text{REBAF} = \langle \mathbf{A}, \mathbf{R}_a, \mathbf{R}_e, \mathbf{s}, \mathbf{t}, \mathbf{P} \rangle$. *Let* $\mathbf{C} = (x_0, \ldots, x_{n-1}, x_n)$ *and* $\mathbf{C}' = (x'_0, \ldots, x'_{m-1}, x'_m)$ *be two DCS of this REBAF such that there exist* $x_i \in \mathbf{C}$ *and* $x'_j \in \mathbf{C}'$ *and* $x_i = x'_j$.

The aggregation *of* $\mathbf{C}$ *and* $\mathbf{C}'$ *is the directed cycle corresponding to the union of the sets* $\{x_0, \ldots, x_{n-1}\}$ *and* $\{x'_0, \ldots, x'_{m-1}\}$. *This aggregation will be denoted by abuse of language* $\mathbf{C} \cup \mathbf{C}'$.

Using this notion of aggregation, a maximal DCS of a REBAF can be defined:

Definition 11 *Let* $\text{REBAF} = \langle \mathbf{A}, \mathbf{R}_a, \mathbf{R}_e, \mathbf{s}, \mathbf{t}, \mathbf{P} \rangle$. *Let* $\mathbf{C} = (x_0, \ldots, x_{n-1}, x_n)$ *be a DCS.* $\mathbf{C}$ *is a* maximal DCS *iff there does not exist another DCS that could be aggregated with* $\mathbf{C}$.

Example 6 (cont'd): In this example, the two simple DCS can be aggregated since they share several elements (β, c, γ, d). And this aggregation is the only maximal DCS of this REBAF:

$$\mathbf{C}'' = (a, \alpha, \beta, c, \gamma, d, \mu, b, \beta, c, \gamma, d, \delta, a)$$

□

3 Background on the Logical Description of a REBAF

Here, we recall the logical description of a REBAF proposed in [11] then improved in [20], that allows an explicit representation of arguments, attacks, *evidential supports* and their properties. In [11] a variant of REBAF has been considered in which interactions are restricted to binary interactions (that is for any interaction α, $\mathbf{s}(\alpha)$ is a singleton) and the support relation is assumed to be acyclic. In [20] the last constraint has been relaxed but not the first one. As a consequence, the definitions of $Def_X(U)$ and $Sup(U)$ given in Definition 5 can be simplified as follows:

Definition 12 *Given a structure* $U = (S, \Gamma, \Delta)$,

- $Def_X(U) \stackrel{\text{def}}{=} \{x \in X | \exists \alpha \in \Gamma, \mathbf{s}(\alpha) \in S \text{ and } \mathbf{t}(\alpha) = x\}$ *with* $X \in \{\mathbf{A}, \mathbf{R}_a, \mathbf{R}_e\}$.
- $Sup(U) \stackrel{\text{def}}{=} \mathbf{P} \cup \{\mathbf{t}(\alpha) | \exists \alpha \in (\Delta \cap Sup(U \backslash \{\mathbf{t}(\alpha)\})), \mathbf{s}(\alpha) \in (S \cap Sup(U \backslash \{\mathbf{t}(\alpha)\}))\}$

3.1 Vocabulary

In [11, 20], the following unary predicate symbols and unary functions symbols are used with the following meaning:

- $Arg(x)$ means "x is an argument"
- $Attack(x)$ means "x is an attack"
- $ESupport(x)$ means that "x is an evidential support"
- $T(x)$ (resp. $S(x)$) denotes the target (resp. source) of x, when x denotes an attack ou a support
- $PrimaFacie(x)$ means that "x is a prima-facie element"
- $Acc(x)$ (resp. $NAcc(x)$) means "x is accepted" (resp. "x cannot be accepted"), when x denotes an argument
- $Val(\alpha)$ means "α is valid" when α denotes an attack or a support

The binary equality predicate is also used. Note that the quantifiers $\exists$ and $\forall$ range over some domain D. To restrict them to subsets of D, bounded quantifiers will be used:
$\forall x \in E\ (P(x))$ means $\forall x\ (x \in E \rightarrow P(x))$ or equivalently $\forall x(E(x) \rightarrow P(x))$.
So we will use:

- $\forall x \in Attack\ (\Phi(x))$ (resp. $\exists x \in Attack\ (\Phi(x))$)
- $\forall x \in ESupport\ (\Phi(x))$ (resp. $\exists x \in ESupport\ (\Phi(x))$)
- and $\forall x \in Arg\ (\Phi(x))$ (resp. $\exists x \in Arg\ (\Phi(x))$).

Note that the meaning of $NAcc(x)$ is not "x is not accepted" but rather "x cannot be accepted" (for instance because x is the target of a valid attack whose source is accepted). Hence, $NAcc(x)$ is not logically equivalent to $\neg Acc(x)$. However, the logical theory will enable to deduce $\neg Acc(x)$ from $NAcc(x)$, as shown below.

Then we need symbols for denoting acceptability of elements. Let us recall that the purpose of [11] was to obtain a logical characterization of structures. As explained before, intuitively, a structure of REBAF represents the set of acceptable arguments (attacks and supports) w.r.t. the structure. And following Definition 6, acceptability w.r.t. a structure requires two conditions, one of them being a support by the structure, the other one making use of the notion of unsupportability. So the following unary predicate symbols are introduced in [11]:

- $Supp$ for denoting supported elements (argument, attack or support),
- $UnSupp$ for denoting unsupportable elements and
- $eAcc$ (resp. $eVal$) for denoting acceptability for arguments (resp. for interactions, attacks or supports).

Note that $eAcc(x)$ ("x is e-accepted") can be understood as "x is accepted and supported" and similarly $eVal(\alpha)$ ("α is e-valid") can be understood as "α is valid and supported".

3.2 Logical theory for describing REBAF

In [11, 20], the formulae describing a given **REBAF** have been partitioned in two sets:

- The first set, denoted by Π, contains the formulae describing the general behaviour of an attack, possibly recursive, *i.e.* how an attack interacts with arguments and other attacks related to it, and also the formulae describing the general behaviour of an evidential support, possibly recursive, *i.e.* how a support interacts with arguments and other interactions related to it.
- The second set, denoted by Π(**REBAF**), contains the formulae encoding the specificities of the current framework.

The meaning of an attack is described under the form of constraints on its source (an argument) and its target (an argument or an attack). Moreover, as attacks may be attacked by other attacks, some attacks may not be valid. And finally supports must be taken into account in order to define this "validity". So we have:

- If an attack from an argument to an attack (or a support) is e-valid, then if its source is e-accepted, its target *is not* valid.
- If an attack between two arguments is e-valid and if its source is e-accepted, then its target *cannot be* accepted. In that case, the target *is not* accepted.

An evidential support can be described by the following constraints:

- If an element (argument or interaction) is prima-facie, it is supported.
- If an element is the target of an evidential support, it is supported if the source of the support is e-accepted and if the support is itself e-valid.

Using the vocabulary defined above,[9] these constraints have been expressed in [11] by the following formulae:

(1) $\forall x \in (Attack \cup ESupport)\ \forall y \in Attack$
$$\left(\begin{array}{l} (eVal(y) \wedge (t_y = x) \wedge eAcc(s_y)) \\ \rightarrow \neg Val(x) \end{array} \right)$$

(2) $\forall x \in Arg\ \forall y \in Attack$
$$\left(\begin{array}{l} (eVal(y) \wedge (t_y = x) \wedge eAcc(s_y)) \\ \rightarrow NAcc(x) \end{array} \right)$$

(3) $\forall x \in Arg\ (NAcc(x) \rightarrow \neg Acc(x))$

(1bis) $\forall x \in (Attack \cup ESupport \cup Arg)$
$$\left(\begin{array}{l} \left(\begin{array}{l} PrimaFacie(x) \vee \\ \exists y \in ESupport \\ \quad (eVal(y) \wedge (t_y = x) \wedge eAcc(s_y)) \end{array} \right) \\ \rightarrow Supp(x) \end{array} \right)$$

The following formulae define the e-acceptability (resp. e-validity). Recall that $eAcc(x)$ (resp. $eVal$) means "x is accepted (resp. valid) *and* supported":

(2bis) $\forall x \in Arg\ ((Acc(x) \wedge Supp(x)) \leftrightarrow eAcc(x))$

(3bis) $\forall x \in (Attack \cup ESupport)$
$((Val(x) \wedge Supp(x)) \leftrightarrow eVal(x))$

Other formulae limit the domain to arguments, attacks, supports.

(4) $\forall x\ (Attack(x) \rightarrow \neg Arg(x))$

(4bis) $\forall x\ (Attack(x) \rightarrow \neg ESupport(x))$

(4ter) $\forall x\ (ESupport(x) \rightarrow \neg Arg(x))$

[9] In the remainder of the paper, we will write s_α (resp. t_α) in place of $S(\alpha)$ (resp. $T(\alpha)$) for simplicity.

(5) $\forall x\ (Arg(x) \vee Attack(x) \vee ESupport(x))$

The logical theory Π consists of all the above formulae.
Then the logical encoding of specificities of a given **REBAF** leads to the set $\Pi(\textbf{REBAF})$ consisting of the following formulae. Let $\mathbf{A} = \{a_1, \ldots a_n\}$, $\mathbf{R}_a = \{\alpha_1, \ldots, \alpha_k\}$, $\mathbf{R}_e = \{\alpha_{k+1}, \ldots, \alpha_m\}$ and $\mathbf{P} = \{x_1, \ldots x_l\}$. [10]

(6) $(s_\alpha = a) \wedge (t_\alpha = b)$ for all $\alpha \in \mathbf{R}_a \cup \mathbf{R}_e$ with $\mathbf{s}(\alpha) = a$ and $\mathbf{t}(\alpha) = b$

(7) $\forall x\ (Arg(x) \leftrightarrow (x = a_1) \vee \ldots \vee (x = a_n))$

(8) $\forall x\ (Attack(x) \leftrightarrow (x = \alpha_1) \vee \ldots \vee (x = \alpha_k))$

(8bis) $\forall x\ (ESupport(x) \leftrightarrow (x = \alpha_{k+1}) \vee \ldots \vee (x = \alpha_m))$

(8ter) $\forall x\ (PrimaFacie(x) \leftrightarrow (x = x_1) \vee \ldots \vee (x = x_l))$

(9) $a_i \neq a_j$ for all $a_i, a_j \in \mathbf{A}$ with $i \neq j$

(10) $\alpha_i \neq \alpha_j$ for all $\alpha_i, \alpha_j \in \mathbf{R}_a \cup \mathbf{R}_e$ with $i \neq j$

Given **REBAF** a higher-order argumentation framework, $\Sigma(\textbf{REBAF})$ will denote the set of first-order logic formulae describing **REBAF**. And so the logical theory $\Sigma(\textbf{REBAF})$ is the union of Π and $\Pi(\textbf{REBAF})$. It is obviously consistent.

In the following examples, using the equality axioms, a simplified version of $\Sigma(\textbf{REBAF})$ is given.[11]

Example 4 (cont'd): Considering the version 1 of this example, we have:

$\Sigma(\textbf{REBAF}) = \{Supp(a)$ (from **(1bis)**, **(8ter)**),
$Supp(\alpha)$ (from **(1bis)**,**(8ter)**),
$(eAcc(a) \wedge eVal(\alpha)) \rightarrow Supp(b)$ (from **(1bis)**),
$(Supp(a) \wedge Acc(a)) \leftrightarrow eAcc(a)$ (from **(2bis)**),
$(Supp(b) \wedge Acc(b)) \leftrightarrow eAcc(b)$ (from **(2bis)**),
$(Supp(\alpha) \wedge Val(\alpha)) \leftrightarrow eVal(\alpha)$ (from **(3bis)**)$\}$

Considering the version 2 of this example, we have:

$\Sigma(\textbf{REBAF}) = \{Supp(a)$ (from **(1bis)**, **(8ter)**),
$(eAcc(a) \wedge eVal(\alpha)) \rightarrow Supp(b)$ (from **(1bis)**),
$(Supp(a) \wedge Acc(a)) \leftrightarrow eAcc(a)$ (from **(2bis)**),
$(Supp(b) \wedge Acc(b)) \leftrightarrow eAcc(b)$ (from **(2bis)**),
$(Supp(\alpha) \wedge Val(\alpha)) \leftrightarrow eVal(\alpha)$ (from **(3bis)**)$\}$

□

Example 2 (cont'd): Note that this example is a variant of the version 1 of Example 4 in which the attack β targeting α has been added.

$\Sigma(\textbf{REBAF}) = \{(eVal(\beta) \wedge eAcc(c)) \rightarrow \neg Val(\alpha)$ (from **(1)**),
$Supp(a)$ (from **(1bis)**, **(8ter)**),
$Supp(c)$ (from **(1bis)**, **(8ter)**),
$Supp(\alpha)$ (from **(1bis)**,**(8ter)**),

[10] We recall that $\mathbf{P} \subseteq \mathbf{A} \cup \mathbf{R}_a \cup \mathbf{R}_e$.
[11] We omit the formulae issued from **(4)** to **(10)** and the tautologies.

$Supp(\beta)$ (from **(1bis)**, **(8ter)**),
$(eAcc(a) \wedge eVal(\alpha)) \rightarrow Supp(b)$ (from **(1bis)**),
$(Supp(a) \wedge Acc(a)) \leftrightarrow eAcc(a)$ (from **(2bis)**),
$(Supp(b) \wedge Acc(b)) \leftrightarrow eAcc(b)$ (from **(2bis)**),
$(Supp(c) \wedge Acc(c)) \leftrightarrow eAcc(c)$ (from **(2bis)**),
$(Supp(\alpha) \wedge Val(\alpha)) \leftrightarrow eVal(\alpha)$ (from **(3bis)**),
$(Supp(\beta) \wedge Val(\beta)) \leftrightarrow eVal(\beta)$ (from **(3bis)**)$\}$

□

3.3 Logical Formalization of REBAF semantics

In presence of higher-order attacks and supports, the conflict-freeness, defence, reinstatement and stability principles must take into account the fact that acceptability for an argument or an interaction requires that any attack against it is unactivable. Moreover acceptability requires support.

3.3.1 Conflict-freeness

In [11, 20], the conflict-freeness principle has been formulated as follows:

- If there is an e-valid attack between two arguments, these arguments cannot be jointly e-accepted.
- If there is an e-valid attack from an e-accepted argument to an interaction (attack or support), this interaction cannot be e-valid.

Note that these properties are already expressed in $\Sigma(\mathbf{REBAF})$ (by the formulae **(1)**, **(2)**, **(3)**, **(2bis)**, **(3bis)**).

3.3.2 Self-supporting

The self-supporting principle states that each supported element must receive evidential support. In [11, 20], it has been formulated as follows:

- If an element is supported then, either it is prima-facie, or it is the target of an e-valid support from an e-accepted source:

(17)

$$\forall x \in (Attack \cup ESupport \cup Arg) \left(\begin{array}{l} Supp(x) \\ \rightarrow \left(\begin{array}{l} PrimaFacie(x) \vee \\ \exists y \in ESupport \\ \quad (eVal(y) \wedge (t_y = x) \wedge eAcc(s_y)) \end{array} \right) \end{array} \right)$$

- Supportability is a weaker notion, as elements that are not supportable (*i.e.* unsupportable) cannot be supported. An element is unsupportable iff it is not prima-facie and for each of its supports, either the support itself or its source is defeated, or the support or its source is in turn unsupportable:

(18)

$$\forall x \in (Attack \cup ESupport \cup Arg) \left(\begin{array}{l} UnSupp(x) \\ \leftrightarrow \left(\begin{array}{l} \neg PrimaFacie(x) \wedge \\ \forall y \in ESupport(t_y = x \\ \quad \rightarrow \left(\begin{array}{l} \exists \beta \in Attack(t_\beta \in \{s_y, y\} \wedge \\ \quad eVal(\beta) \wedge eAcc(s_\beta))) \\ \vee UnSupp(s_y) \\ \vee UnSupp(y)) \end{array} \right) \end{array} \right) \end{array} \right)$$

Formulae **(17)** and **(18)** are added to the base $\Sigma(\mathbf{REBAF})$, thus producing the base $\Sigma_{ss}(\mathbf{REBAF})$.

3.3.3 Defence

As stated in Definition 6, an attacked element is acceptable if *(i)* it is supported and *(ii)* for each attack against it, either the source or the attack itself is defeated (by an e-valid attack from an e-accepted argument), or the source or the attack itself is unsupportable (w.r.t. e-valid elements and e-accepted arguments).
So, in [11, 20], the principle corresponding to the previous item *(ii)* has been expressed by the following formulae that are associated with formulae **(17)** and **(18)**:

(11)

$$\forall \alpha \in Attack \left(\begin{array}{l} Acc(t_\alpha) \\ \rightarrow \left(\begin{array}{l} \exists \beta \in Attack \\ \quad (t_\beta \in \{s_\alpha, \alpha\} \wedge eVal(\beta) \wedge eAcc(s_\beta)) \\ \vee UnSupp(s_\alpha) \\ \vee UnSupp(\alpha) \end{array} \right) \end{array} \right)$$

(12)

$$\forall \alpha \in Attack\, \forall \delta \in (Attack \cup ESupport) \left(\begin{array}{l} ((\delta = t_\alpha) \wedge Val(\delta)) \\ \rightarrow \left(\begin{array}{l} \exists \beta \in Attack \\ \quad (t_\beta \in \{s_\alpha, \alpha\} \wedge eVal(\beta) \wedge eAcc(s_\beta)) \\ \vee UnSupp(s_\alpha) \\ \vee UnSupp(\alpha) \end{array} \right) \end{array} \right)$$

Formulae **(11)** and **(12)** are added to the base $\Sigma_{ss}(\mathbf{REBAF})$, thus producing the base $\Sigma_{d}(\mathbf{REBAF})$.

3.3.4 Reinstatement

In [11, 20], the reinstatement principle has been expressed by the following formulae that are be associated with formulae **(17)** and **(18)**:

(13)

$$\forall c \in Arg \left(\begin{array}{l} \left(\begin{array}{l} \forall \alpha \in Attack \\ \left(\begin{array}{l} t_\alpha = c \rightarrow \\ \left(\begin{array}{l} \exists \beta \in Attack (t_\beta \in \{s_\alpha, \alpha\} \wedge \\ \quad eVal(\beta) \wedge eAcc(s_\beta)) \\ \vee UnSupp(s_\alpha) \\ \vee UnSupp(\alpha) \end{array} \right) \end{array} \right) \end{array} \right) \\ \rightarrow Acc(c) \end{array} \right)$$

(14)

$$\forall \delta \in (Attack \cup ESupport) \left(\begin{array}{l} \left(\begin{array}{l} (\forall \alpha \in Attack \\ \left(\begin{array}{l} t_\alpha = \delta \rightarrow \\ \left(\begin{array}{l} \exists \beta \in Attack (t_\beta \in \{s_\alpha, \alpha\} \wedge \\ \quad eVal(\beta) \wedge eAcc(s_\beta)) \\ \vee UnSupp(s_\alpha) \\ \vee UnSupp(\alpha) \end{array} \right) \end{array} \right) \end{array} \right) \\ \rightarrow Val(\delta) \end{array} \right)$$

Formulae **(13)** and **(14)** are added to the base $\Sigma_{ss}(\textbf{REBAF})$, thus producing the base $\Sigma_r(\textbf{REBAF})$.

3.3.5 Stability

In [11,20], the stability principle has been expressed by the three following formulae that are associated with formulae **(17)** and **(18)**:[12]

(15) $\forall c \in Arg$
$$\left(\begin{array}{l} \neg Acc(c) \\ \rightarrow \left(\begin{array}{l} \exists \beta \in Attack(t_\beta = c \wedge \\ \quad eVal(\beta) \wedge eAcc(s_\beta)) \end{array} \right) \end{array} \right)$$

(16) $\forall \alpha \in (Attack \cup ESupport)$
$$\left(\begin{array}{l} \neg Val(\alpha) \\ \rightarrow \left(\begin{array}{l} \exists \beta \in Attack(t_\beta = \alpha \wedge \\ \quad eVal(\beta) \wedge eAcc(s_\beta)) \end{array} \right) \end{array} \right)$$

(19) $\forall x \in (Arg \cup Attack \cup ESupport)$
$$(\neg Supp(x) \rightarrow UnSupp(x))$$

Formulae **(15)**, **(16)** and **(19)** are added to the base $\Sigma_{ss}(\textbf{REBAF})$, thus producing the base $\Sigma_s(\textbf{REBAF})$.

Example 4 (cont'd): Considering the version 1, $\Sigma_{ss}(\textbf{REBAF})$ is obtained from $\Sigma(\textbf{REBAF})$ by adding the following formulae:

$Supp(b) \rightarrow (eAcc(a) \wedge eVal(\alpha))$
$\neg UnSupp(a)$
$\neg UnSupp(\alpha)$
$Unsupp(b) \leftrightarrow (UnSupp(a) \vee UnSupp(\alpha))$

As there is no attack, $\Sigma_d(\textbf{REBAF})$ contains nothing more than $\Sigma_{ss}(\textbf{REBAF})$.
And finally $\Sigma_r(\textbf{REBAF})$ is obtained from $\Sigma_{ss}(\textbf{REBAF})$ by adding the formulae: $Acc(a)$, $Acc(b)$ and $Val(\alpha)$.

Considering the version 2, $\Sigma_{ss}(\textbf{REBAF})$ is obtained from $\Sigma(\textbf{REBAF})$ by adding the following formulae:

$Supp(b) \rightarrow (eAcc(a) \wedge eVal(\alpha))$
$\neg Supp(\alpha)$
$\neg UnSupp(a)$
$UnSupp(\alpha)$
$Unsupp(b) \leftrightarrow (UnSupp(a) \vee UnSupp(\alpha))$

Once again, $\Sigma_d(\textbf{REBAF})$ contains nothing more than $\Sigma_{ss}(\textbf{REBAF})$.
And $\Sigma_r(\textbf{REBAF})$ is obtained from $\Sigma_{ss}(\textbf{REBAF})$ by adding the formulae: $Acc(a)$, $Acc(b)$ and $Val(\alpha)$. □

Example 2 (cont'd): $\Sigma_{ss}(\textbf{REBAF})$ is obtained from $\Sigma(\textbf{REBAF})$ by adding formulae among which:

$Supp(b) \rightarrow (eAcc(a) \wedge eVal(\alpha))$
$\neg UnSupp(a)$
$\neg UnSupp(c)$
$\neg UnSupp(\alpha)$
$\neg UnSupp(\beta)$
$$Unsupp(b) \leftrightarrow \left(\begin{array}{l} (eVal(\beta) \wedge eAcc(c)) \\ \vee UnSupp(a) \\ \vee UnSupp(\alpha) \end{array} \right)$$

Then $\Sigma_d(\textbf{REBAF})$ is obtained from $\Sigma_{ss}(\textbf{REBAF})$ by adding formulae among which:

$Val(\alpha) \rightarrow (UnSupp(\beta) \vee UnSupp(c))$

[12] Let us recall that a stable structure $U = (S, \Gamma, \Delta)$ satisfies: $\overline{S \cup \Gamma \cup \Delta} \subseteq UnAcc(U)$.

$\Sigma_r(\mathbf{REBAF})$ is obtained from $\Sigma_{ss}(\mathbf{REBAF})$ by adding the formulae:
$Acc(a)$
$Acc(b)$
$Acc(c)$
$Val(\beta)$
$(UnSupp(c) \vee UnSupp(\beta)) \rightarrow Val(\alpha)$
$\Sigma_s(\mathbf{REBAF})$ is obtained from $\Sigma_{ss}(\mathbf{REBAF})$ by adding the formulae:
$Acc(a)$
$Acc(b)$
$Acc(c)$
$Val(\beta)$
$\neg Val(\alpha) \rightarrow eVal(\beta) \wedge eAcc(c)$
$\neg Supp(b) \rightarrow UnSupp(b)$ and also
$\neg Supp(x) \rightarrow UnSupp(x)$ for $x \in \{a, c, \alpha, \beta\}$ □

3.4 Characterizing Semantics of a REBAF

[11] proposed characterizations of the REBAF structures under different semantics in terms of models of the bases $\Sigma(\mathbf{REBAF})$, $\Sigma_d(\mathbf{REBAF})$, $\Sigma_r(\mathbf{REBAF})$, $\Sigma_s(\mathbf{REBAF})$. The common idea is that a structure gathers the acceptable elements w.r.t. it.

Let $\mathbf{REBAF} = \langle \mathbf{A}, \mathbf{R}_a, \mathbf{R}_e, \mathbf{s}, \mathbf{t}, \mathbf{P} \rangle$. Given $\mathcal{I}$ an interpretation of $\Sigma(\mathbf{REBAF})$, we define:

- $S_\mathcal{I} = \{x \in \mathbf{A} | \mathcal{I}(eAcc(x)) = \textit{true}\}$
- $\Gamma_\mathcal{I} = \{x \in \mathbf{R}_a | \mathcal{I}(eVal(x)) = \textit{true}\}$
- $\Delta_\mathcal{I} = \{x \in \mathbf{R}_e | \mathcal{I}(eVal(x)) = \textit{true}\}$

Moreover, let $\mathcal{I}$ be a model of $\Sigma(\mathbf{REBAF})$:

- $\mathcal{I}$ is a $\subseteq$-maximal model of $\Sigma(\mathbf{REBAF})$ iff there is no model $\mathcal{I}'$ of $\Sigma(\mathbf{REBAF})$ with $(S_\mathcal{I} \cup \Gamma_\mathcal{I} \cup \Delta_\mathcal{I}) \subset (S_{\mathcal{I}'} \cup \Gamma_{\mathcal{I}'} \cup \Delta_{\mathcal{I}'})$.
- $\mathcal{I}$ is a $\subseteq$-minimal model of $\Sigma(\mathbf{REBAF})$ iff there is no model $\mathcal{I}'$ of $\Sigma(\mathbf{REBAF})$ with $(S_{\mathcal{I}'} \cup \Gamma_{\mathcal{I}'} \cup \Delta_{\mathcal{I}'}) \subset (S_\mathcal{I} \cup \Gamma_\mathcal{I} \cup \Delta_\mathcal{I})$.

Let recall that the characterization proposed in [11] applies to a restricted variant of REBAF in which two constraints are given: first *interactions are assumed to be binary* and secondly *there is no cycle of supports*. This second restriction has been relaxed in [20] using the notion of impacting support chains and the fact that the existence of support cycles has an impact on the $UnSupp$ predicate. That leads to the following notion:[13]

Definition 13 *Let* $\mathbf{REBAF} = \langle \mathbf{A}, \mathbf{R}_a, \mathbf{R}_e, \mathbf{s}, \mathbf{t}, \mathbf{P} \rangle$. *$\mathcal{I}$ is a* support-founded interpretation *iff the two following conditions hold:*

1. *for each argument (resp. support) x non prima-facie, belonging to a maximal DCS and such that $\mathcal{I}(eAcc(x)) =$ true (resp. $\mathcal{I}(eVal(x)) =$ true), there exists at least one impacting support chain $\mathbf{ISC} = (x_0, \ldots, x_n)$ for x that is satisfied by $\mathcal{I}$, i.e. $\forall x_i \in \mathbf{ISC}$, if $x_i \in \mathbf{A}$ then $\mathcal{I}(eAcc(x_i)) =$ true, otherwise $\mathcal{I}(eVal(x_i)) =$ true;*
2. *for each element x of* $\mathbf{REBAF}$, *$\mathcal{I}(UnSupp(x)) =$ true iff $x \in UnSupp(U_\mathcal{I})$ with $U_\mathcal{I} = (S_\mathcal{I}, \Gamma_\mathcal{I}, \Delta_\mathcal{I})$.*

Let Σ_x be a base of formulae built over $\mathbf{REBAF}$. *A* support-founded model *of Σ_x is a support-founded interpretation which is a model of Σ_x.*

[13] This is a correction of the notion of support-founded model proposed initially in [11] that has been proved incorrect in [20].

Then using these support-founded models, the following characterization of REBAF semantics is given in [20]:

Proposition 1 *Let* $\mathbf{REBAF} = \langle \mathbf{A}, \mathbf{R}_a, \mathbf{R}_e, \mathbf{s}, \mathbf{t}, \mathbf{P} \rangle$. *Let* $U = (S, \Gamma, \Delta)$ *be a structure on* $\mathbf{REBAF}$.

1. U *is admissible iff there exists* $\mathcal{I}$ *support-founded model of* $\Sigma_d(\mathbf{REBAF})$ *(in the sense of Definition 13) with* $S_\mathcal{I} = S$, $\Gamma_\mathcal{I} = \Gamma$ *and* $\Delta_\mathcal{I} = \Delta$.
2. U *is complete iff there exists* $\mathcal{I}$ *support-founded model of the union* $(\Sigma_d(\mathbf{REBAF}) \cup \Sigma_r(\mathbf{REBAF}))$ *(in the sense of Definition 13) with* $S_\mathcal{I} = S$, $\Gamma_\mathcal{I} = \Gamma$ *and* $\Delta_\mathcal{I} = \Delta$.
3. U *is a preferred structure iff there exists* $\mathcal{I}$ $\subseteq$*-maximal support-founded model of* $\Sigma_d(\mathbf{REBAF})$ *(in the sense of Definition 13) with* $S_\mathcal{I} = S$, $\Gamma_\mathcal{I} = \Gamma$ *and* $\Delta_\mathcal{I} = \Delta$.
4. U *is the grounded structure iff* $S = S_\mathcal{I}$, $\Gamma_\mathcal{I} = \Gamma$ *and* $\Delta_\mathcal{I} = \Delta$ *where* $\mathcal{I}$ *is a* $\subseteq$*-minimal support-founded model of* $(\Sigma_d(\mathbf{REBAF}) \cup \Sigma_r(\mathbf{REBAF}))$ *(in the sense of Definition 13).*
5. U *is stable iff there exists* $\mathcal{I}$ *support-founded model of* $\Sigma_s(\mathbf{REBAF})$ *(in the sense of Definition 13) with* $S_\mathcal{I} = S$, $\Gamma_\mathcal{I} = \Gamma$ *and* $\Delta_\mathcal{I} = \Delta$.

Let us illustrate the above results on the previous examples:

Example 5 (cont'd): Apply Proposition 1 leads to the unique complete, preferred, stable and grounded structure $(\{a, b, c, d, e\}, \varnothing, \{\alpha, \beta, \delta, \gamma, \pi\})$. □

Example 6 (cont'd): Apply Proposition 1 leads to the unique complete, preferred, stable and grounded structure $(\{a, b, c, d, e\}, \varnothing, \{\alpha, \beta, \gamma, \delta, \pi, \mu\})$. □

4 Collective interactions in a REBAF: a new proposition

Considering the logical translation of a REBAF, it remains a constraint that is relaxed neither in [11], nor in [20]: the interactions must be binary ones. So in this section we propose a new encoding that relaxes this constraint and allows collective interactions.

4.1 Vocabulary

The first evolution of the previous encoding is a mandatory modification of the vocabulary in order to take into account the fact that the source of an interaction can be a set of arguments. So the old unary function S becomes now a binary predicate:

$S(a, \alpha)$ means that "the argument a belongs to the source of α"

4.2 Formulae

The second evolution concerns the formulae in which sources appear. Three cases occur and each case corresponds to a particular behaviour:[14]

- The source is used as a parameter in the predicate $eAcc$; in this case, the idea is that the source of an interaction is e-accepted iff *all the arguments* belonging to this source are also e-accepted; so the old formula $eAcc(s_\alpha)$ corresponds to:
$$\forall a \in Arg(S(a, \alpha) \rightarrow eAcc(a))$$
- The source is used as a parameter in the predicate $UnSupp$; in this case, the idea is that the source of an interaction is unsupportable iff *at least one argument* belonging to this source is also unsupportable; so the old formula $UnSupp(s_\alpha)$ corresponds to:
$$\exists a \in Arg(S(a, \alpha) \wedge UnSupp(a))$$

[14]Note that a formula as $x \in \{s_y, y\}$ is just a shortcut for $(x = s_y) \vee (x = y)$ (see for instance formulae **(18)** and **(11)** to **(14)**).

- The source is used as a parameter in the equality predicate; here two subcases are possible depending of the sense of this equality:

  - either the old equality $s_\alpha = a$ becomes a *logical or* between all the elements of the source:
    $$(a_1 = a) \vee (a_2 = a), \vee \ldots \vee (a_n = a), \text{ for } \mathbf{s}(\alpha) = \{a_1, a_2, \ldots, a_n\}$$
    that is equivalent to:
    $$\exists x \in Arg(S(x, \alpha) \wedge x = a)$$
  - or the old equality $s_\alpha = a$ becomes a *logical and* between all the elements of the source:
    $$S(a_1, \alpha) \wedge S(a_2, \alpha), \wedge \ldots \wedge S(a_n, \alpha), \text{ for } \mathbf{s}(\alpha) = \{a_1, a_2, \ldots, a_n\}$$

Of course formulae **(1)** to **(19)** could be rewritten using the new formalism but the result becomes hard to read. So, our choice is to keep the old formulae in which the predicates applied to a source are considered as the shortcut defined as previously.

For instance, for formula **(2)**, we keep:

$$\forall x \in Arg\ \forall y \in Attack \left(\begin{array}{l} (eVal(y) \wedge (t_y = x) \wedge eAcc(s_y)) \\ \rightarrow NAcc(x) \end{array} \right)$$

but that means:

$$\forall x \in Arg\ \forall y \in Attack \left(\begin{array}{l} (eVal(y) \wedge (t_y = x) \wedge \forall x_i \in Arg(S(x_i, y) \rightarrow eAcc(x_i))) \\ \rightarrow NAcc(x) \end{array} \right)$$

Another example is formula **(11)**, we keep:

$$\forall \alpha \in Attack \left(\begin{array}{l} Acc(t_\alpha) \\ \rightarrow \left(\begin{array}{l} \exists \beta \in Attack \\ \quad (t_\beta \in \{s_\alpha, \alpha\} \wedge eVal(\beta) \wedge eAcc(s_\beta)) \\ \vee UnSupp(s_\alpha) \\ \vee UnSupp(\alpha) \end{array} \right) \end{array} \right)$$

but that means:

$$\forall \alpha \in Attack \left(\begin{array}{l} Acc(t_\alpha) \\ \rightarrow \left(\begin{array}{l} \exists \beta \in Attack \\ \quad (\exists x \in Arg(S(x, \alpha) \wedge t_\beta = x) \vee t_\beta = \alpha) \wedge eVal(\beta) \wedge \forall x \in Arg(S(x, \beta) \rightarrow eAcc(x)) \\ \vee \exists x \in Arg(S(x, \alpha) \wedge UnSupp(x)) \\ \vee UnSupp(\alpha) \end{array} \right) \end{array} \right)$$

The only exception of this use of shortcut is formula **(6)** that is very specific and must be rewritten; the old formula
$$(s_\alpha = a) \wedge (t_\alpha = b) \text{ for all } \alpha \in \mathbf{R}_a \cup \mathbf{R}_e \text{ with } \mathbf{s}(\alpha) = a \text{ and } \mathbf{t}(\alpha) = b$$
becomes:
$$S(a_1, \alpha) \wedge \ldots \wedge S(a_n, \alpha) \wedge (t_\alpha = b) \text{ for all } \alpha \in \mathbf{R}_a \cup \mathbf{R}_e \text{ with } \mathbf{s}(\alpha) = \{a_1, \ldots, a_n\} \text{ and } \mathbf{t}(\alpha) = b$$

Note that the definition of the bases of formulae remains unchanged.

At this point, it is worth to note that if there is no support cycles in the REBAF then the use of the formulae bases is enough for characterizing the REBAF semantics since any element could be supported without itself. So the difficulty comes with the existence of these cycles and implies that we are able to remove the models in which an element is supported because it is satisfied by these models. The detection of such models was exactly the aim of Definition 13.

Nevertheless, since we now want to take into account collective supports, some questions appear:

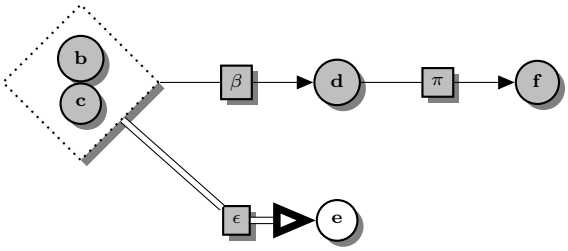
- "Is Definition 13 enough for characterizing support-founded models when collective supports exist in the REBAF?"
- and more generally "How to take into account support cycles when we have collective supports?".

The answer to these questions probably implies new definitions for the notions of DCS, of impacting support chains and of support-founded interpretations.

4.3 Some examples

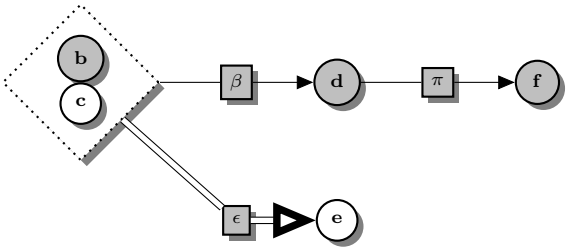
First consider some examples that show how the source of a collective interaction that is not reduced to a singleton can impact the computation of structures (the three first examples have no cycle whereas there are support cycles in the two last ones).

Example 7 *In this example, there are a collective attack and a collective support using the same source (this source is graphically represented by a "dotted diamond" containing the elements composing the source). And e is the only element that is not prima-facie.*



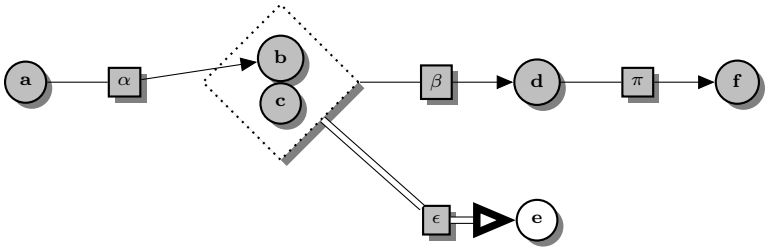
In this case, the source of β and ϵ is composed by supported and not defeated arguments. So β and ϵ are activable and there is one preferred structure that is: $(\{b, c, e, f\}, \{\beta, \pi\}, \{\epsilon\})$.

Example 8 *This is the same example as Example 7, except that argument c is now not prima-facie.*



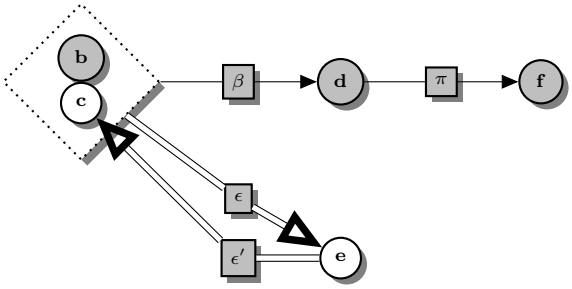
Since c is unsupportable, then neither β nor ϵ can be activable and there is one preferred structure that is: $(\{b, d\}, \{\beta, \pi\}, \{\epsilon\})$.

Example 9 *This example is an extension of Example 7 in which we add an attack against one argument of the source of the collective interactions.*



Here, b is attacked and cannot be defended. So β and ϵ are unactivable and there is one preferred structure that is: $(\{a, c, d\}, \{\alpha, \beta, \pi\}, \{\epsilon\})$.

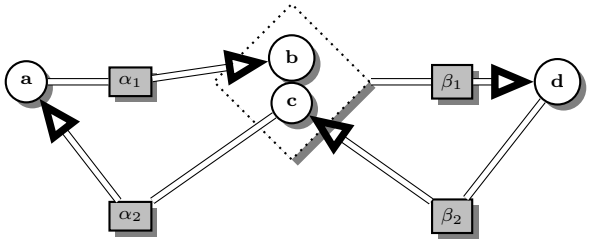
Example 10 *This is an extension of Example 8 with an additional support from e to c.*



Since c cannot be supported without e and e cannot be supported without $\{b, c\}$, a support cycle appears. So neither β nor ϵ can be activable and there is one preferred structure that is: $(\{b, d\}, \{\beta, \pi\}, \{\epsilon, \epsilon'\})$.

The interesting point is the form of this support cycle that is slightly different that the description given in Definition 8, since the target of ϵ' (i.e. c) and the source of ϵ (i.e. $\{b, c\}$) do not coincide whereas ϵ follows ϵ' in the cycle.

Example 11 *This example gives another example of support cycles.*



In this example, none argument is prima-facie. Moreover, they are unsupportable since they cannot be supported without themselves: a is supported if c is supported; c is supported if d is supported; d is supported if $\{b, c\}$ is supported, so if b and c are supported; and finally b is supported if a is supported. Moreover it graphically seems that there are here several supports cycles that are interconnected.

Here the only one preferred structure that is: $(\varnothing, \varnothing, \{\alpha_1, \alpha_2, \beta_1, \beta_2\})$.

4.4 New definitions

In this section we adapt the previous definitions about support cycles in order to take into account collective interactions.

The first definition that must be adapted is the definition of DCS. Here the important point is the fact that, in a cycle, the targets and the sources must be clearly identified:

Definition 14 *Let* REBAF $= \langle \mathbf{A}, \mathbf{R}_a, \mathbf{R}_e, \mathbf{s}, \mathbf{t}, \mathbf{P} \rangle$. *A* directed cycle of supports (DCS) *in this REBAF is a sequence* $\mathbf{C} = (x_0, \ldots, x_{n-1}, x_n)$ *such that:*[15]

- $n > 0$ *and* n *is the* size of the DCS*,*
- $\forall i = 0 \ldots n$*, either* $x_i \in \mathbf{R}_e$*, or* $x_i = (a, S)$ *with* $S \in 2^{\mathbf{A}} \setminus \varnothing$ *and* $a \in \mathbf{A} \cap S$ *(a is called the "target field" of x_i and S is called the "source field" of x_i),*
- $x_n = x_0$

[15] By abuse of language, the set of the elements composing $\mathbf{C}$ will be also denoted by $\mathbf{C}$. So $\mathbf{C}$ will be used with set operators as $\cap$ ou $\cup$ and will be comparable with other sets.

- $\forall i = 0 \dots n-1$, *if* $x_i = (a, S) \in (\mathbf{A}, 2^{\mathbf{A}} \setminus \varnothing)$ *then* $x_{i+1} \in \mathbf{R}_e$ *and* $\mathbf{s}(x_{i+1}) = S$,
- $\forall i = 0 \dots n-1$, *if* $x_i \in \mathbf{R}_e$ *then*
  - *if* $x_{i+1} \in \mathbf{R}_e$ *then* $\mathbf{t}(x_i) = x_{i+1}$
  - *if* $x_{i+1} = (a, S) \in (\mathbf{A}, 2^{\mathbf{A}} \setminus \varnothing)$ *then* $\mathbf{t}(x_i) = a$.

A simple DCS $\mathbf{C} = (x_0, \dots, x_{n-1}, x_n)$ *is a DCS in which* $\forall i, j = 0 \dots n-1$, *if* $i \neq j$ *then* $x_i \neq x_j$.
An input support of a DCS $\mathbf{C} = (x_0, \dots, x_{n-1}, x_n)$ *is:*

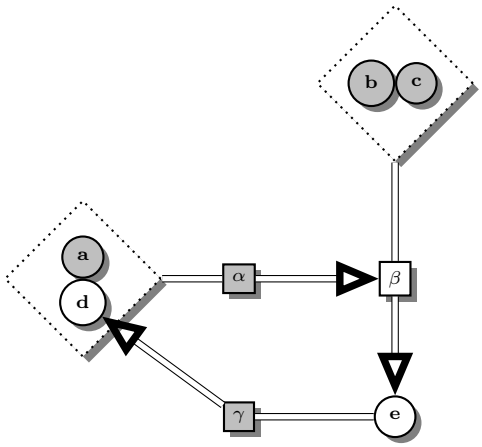
- *either a support* $y \in \mathbf{R}_e \setminus \mathbf{C}$ *and* $\exists x_i \in \mathbf{C}$ *such that:*
  - *if* $x_i \in \mathbf{R}_e$ *then* $\mathbf{t}(y) = x_i$,
  - *if* $x_i = (a, S) \in (\mathbf{A}, 2^{\mathbf{A}} \setminus \varnothing)$ *then* $\mathbf{t}(y) = a$,
- *or a set of arguments* $y \in 2^{\mathbf{A}} \setminus \varnothing$ *such that* $y \notin \mathbf{C}$ *and* $\exists x_i \in \mathbf{R}_e \cap \mathbf{C}$ *and* $y = \mathbf{s}(x_i)$.

The set of inputs of the DCS $\mathbf{C}$ *is denoted by* $\mathbf{C}^{In}$ *and it is partitioned into* $\mathbf{C}^{In}_{\mathbf{A}} = \mathbf{C}^{In} \cap (\mathbf{A}, 2^{\mathbf{A}} \setminus \varnothing)$ *and* $\mathbf{C}^{In}_{\mathbf{R}_e} = \mathbf{C}^{In} \cap \mathbf{R}_e$.

Note that a DCS is now an "hybrid" sequence composed either with interactions, or with pairs (an argument, a no empty set of arguments). The other definitions (for aggregation and maximal DSC) remain unchanged.

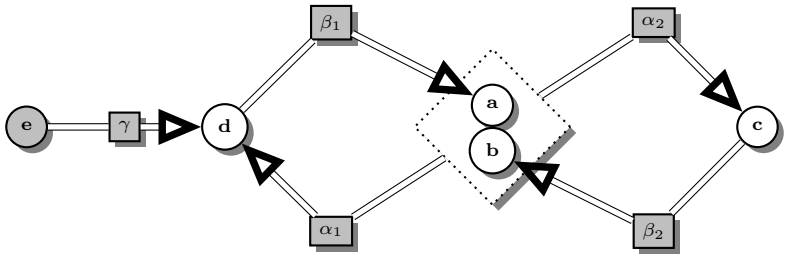
Example 10 (cont'd): In this example, there is only one DCS whose size is 4: $((c, \{b, c\}), \epsilon, (e, \{e\}), \epsilon', (c, \{b, c\}))$.
Note that a DCS whose size is n can be represented by n different sequences obtained by a shift to the right or to the left. For instance, in this example, the DCS can also be written as: $\epsilon, (e, \{e\}), \epsilon', (c, \{b, c\}), \epsilon)$ □

Example 12 *This example gives another example of support cycles with an higher-order support. Here* β, d *and* e *are not prima-facie.*



Here there is one DCS: $((d, \{a, d\}), \alpha, \beta, (e, \{e\}), \gamma, (d, \{a, d\}))$.
Note that the set $\{b, c\}$ *is an input of this DCS; moreover, the only preferred structure is* $(\{a, b, c\}, \varnothing, \{\alpha, \gamma\})$.

Example 13 *This example gives an example of several support cycles that can be aggregated. Here the only preferred structure is* $(\{e, d, a\}, \varnothing, \{\alpha_1, \alpha_2, \beta_1, \beta_2, \gamma\})$.

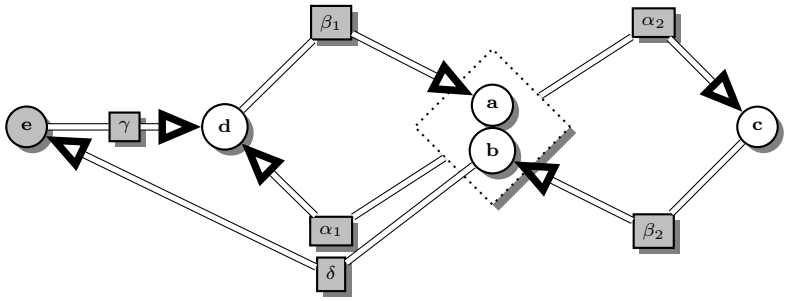


Here there are three DCS (only the last one is a maximal DCS):

- $((d, \{d\}), \beta_1, (a, \{a, b\}), \alpha_1, (d, \{d\}))$
- $((c, \{c\}), \beta_2, (b, \{a, b\}), \alpha_2, (c, \{c\}))$
- $((d, \{d\}), \beta_1, (a, \{a, b\}), \alpha_2, (c, \{c\}), \beta_2, (b, \{a, b\}), \alpha_1, (d, \{d\}))$

The interesting point is the fact that the set $\{a, b\}$ that is the source of α_1 and α_2 corresponds to two distinct elements in a DCS: $(a, \{a, b\})$ and $(b, \{a, b\})$; and each of them can be used as the preceding element of the supports α_1 or α_2 in the DCS.

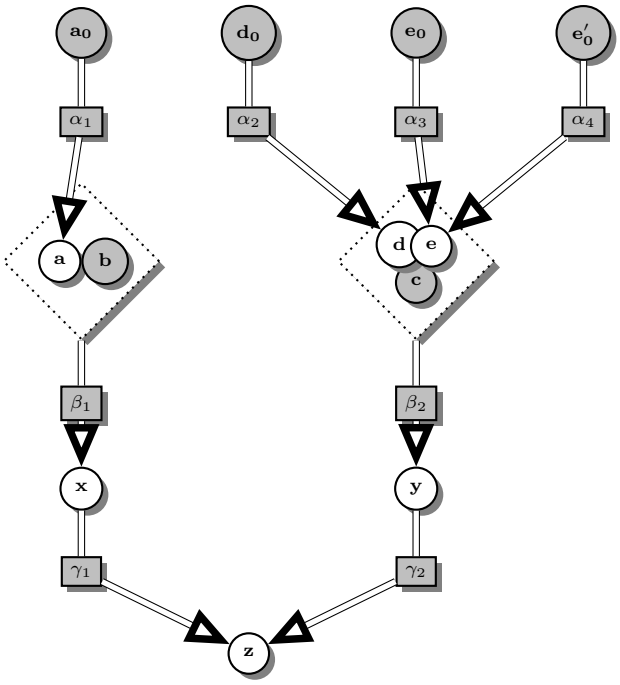
Moreover it could be possible to also have $(b, \{b\})$ in a DCS since, for instance a support exists in the REBAF using b as source. Consider for instance the following REBAF:



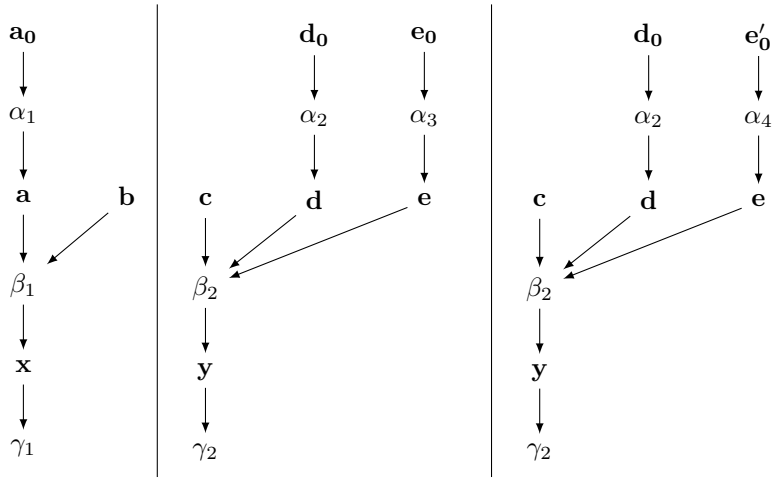
In this case, the maximal DCS is: $((d, \{d\}), \beta_1, (a, \{a, b\}), \alpha_2, (c, \{c\}), \beta_2, (b, \{b\}), \delta, (e, \{e\}), \gamma, (d, \{d\}), \beta_1, (a, \{a, b\}), \alpha_1, (d, \{d\}))$.

Consider now the notion of impacting support chain. The following example shows that this notion must also be improved:

Example 14



Let consider now the elements that impact the supported status of argument z. Clearly simple chains are not enough and we must use the notion of "trees"; indeed, any element of the source of a collective support must be supported if we want the target of this support to be also supported. Here, three "trees" must be taken into account for computing the supported status of z.



The previous example gives the main ideas for defining the notion of impacting support tree for an element of the REBAF:

Definition 15 *Let* $\mathbf{REBAF} = \langle \mathbf{A}, \mathbf{R}_a, \mathbf{R}_e, \mathbf{s}, \mathbf{t}, \mathbf{P} \rangle$. *Let x be an element of this REBAF. An* impacting support tree *for x is a set* $\mathbf{IST} = \{x_0, \ldots, x_n\}$ *with $n > 0$ defined as follows:*

- $\forall x_i, i \in [0 \ldots n]$, $x_i \in (\mathbf{A} \cup \mathbf{R}_e) \setminus \{x\}$ *and is called a* node *of the tree;*
- *Let* $\mathbf{IST_P} = (\mathbf{IST} \cap \mathbf{P} \cap \mathbf{A})$. $\mathbf{IST_P} \neq \varnothing$*;*

- $!\exists x_i \in \mathbf{IST}$ *such that* $x_i \in \mathbf{R}_e$ *and* $\mathbf{t}(x_i) = x$*; this* x_i *is called the* root *of the tree;*
- $\forall i, j \in [0 \dots n]$*, if* $i \neq j$*, then* $x_i \neq x_j$*;*
- $\forall x_i \in \mathbf{IST} \cap \mathbf{A}$*, either* $\exists x_j \in \mathbf{IST} \cap \mathbf{R}_e$ *such that* $x_i = \mathbf{t}(x_j)$*, or* $x_i \in \mathbf{IST_P}$ *(in this case* x_i *is called a* leaf *of the tree);*
- $\forall x_i \in \mathbf{IST} \cap \mathbf{R}_e$, $\forall x_j \in \mathbf{s}(x_i)$, $x_j \in \mathbf{IST}$.

Note that, as in Definition 9, an element x cannot belong to its impacting support tree, and by definition non repetition is authorized.

Example 7 (cont'd): There is no $\mathbf{IST}$ for any interaction or argument except for e. And the $\mathbf{IST}$ for e is: $\{b, c, \epsilon\}$, with ϵ that is the unique root of $\mathbf{IST}$ and b, c that are the leaves of the tree. Note that $\mathbf{IST_P} = \{b, c\}$ and so is not empty. □

Example 8 (cont'd): In this example, even e has no $\mathbf{IST}$. Indeed the support ϵ cannot belong to an $\mathbf{IST}$ for e since one argument of its source (c) cannot belong to an $\mathbf{IST}$ (it is neither prima-facie, nor targeted by a support). □

Example 9 (cont'd): In this case, we obtain the same result as in Example 7: there is no $\mathbf{IST}$ for any interaction or argument except for e and the $\mathbf{IST}$ for e is: $\{b, c, \epsilon\}$.

Note that the fact that b is attacked but not defended has no impact on the building of an $\mathbf{IST}$ containing b. Indeed the existence of an $\mathbf{IST}$ for an element x is not a guarantee for the supportability of x. It is just a necessary condition. □

Example 10 (cont'd): In this example, due to the existence of a DCS, there is no $\mathbf{IST}$ for e, since, by definition, e cannot belong to an $\mathbf{IST}$ for itself. □

Example 11 (cont'd): Here arguments have no $\mathbf{IST}$ (because of the existence of a DCS) and interactions have no $\mathbf{IST}$ (because they are not targeted by a support). □

Example 12 (cont'd): Considering the existence of the DCS and the fact that d in this DCS needs to be supported without itself, there is no $\mathbf{IST}$ for d and so for β, and e. □

Example 14 (cont'd): Considering a, there is one $\mathbf{IST} = \{\alpha_1, a_0\}$. Considering d, there is one $\mathbf{IST} = \{\alpha_2, d_0\}$. Considering e, there are two $\mathbf{IST}$, $\{\alpha_3, e_0\}$ and $\{\alpha_4, e'_0\}$. Considering z, there are three $\mathbf{IST}$:

- $\{\gamma_1, x, \beta_1, a, b, \alpha_1, a_0\}$ (γ_1 being the root, a_0 and b being the leaves),
- $\{\gamma_2, y, \beta_2, c, d, e, \alpha_2, d_0, \alpha_3, e_0\}$ (γ_2 being the root, c, d_0 and e_0 being the leaves),
- $\{\gamma_2, y, \beta_2, c, d, e, \alpha_2, d_0, \alpha_4, e'_0\}$ (γ_2 being the root, c, d_0 and e'_0 being the leaves).

The other elements of the REBAF have no $\mathbf{IST}$. □

4.5 Characterization: a new proposition

The previous definition completed by the constraint concerning the unsupportable status of the element[16] leads to the following new definition for support-founded interpretations and models:

Definition 16 *Let* $\mathbf{REBAF} = \langle \mathbf{A}, \mathbf{R}_a, \mathbf{R}_e, \mathbf{s}, \mathbf{t}, \mathbf{P} \rangle$*.* $\mathcal{I}$ *is a* support-founded interpretation *iff the two following conditions hold:*

1. *for each argument (resp. support)* x *non prima-facie, belonging to a maximal DCS and such that* $\mathcal{I}(eAcc(x)) =$ *true (resp.* $\mathcal{I}(eVal(x)) =$ *true), there exists at least one impacting support tree* $\mathbf{IST} = (x_0, \dots, x_n)$ *for* x *that is satisfied by* $\mathcal{I}$*, i.e.* $\forall x_i \in \mathbf{IST}$*, if* $x_i \in \mathbf{A}$ *then* $\mathcal{I}(eAcc(x_i)) =$ *true, otherwise* $\mathcal{I}(eVal(x_i)) =$ *true;*

[16] See Definition 9.

2. *for each element* x *of* **REBAF**, $\mathcal{I}(UnSupp(x)) =$ *true iff* $x \in UnSupp(U_{\mathcal{I}})$ *with* $U_{\mathcal{I}} = (S_{\mathcal{I}}, \Gamma_{\mathcal{I}}, \Delta_{\mathcal{I}})$.

Let Σ_x *be a base of formulae built over* **REBAF**. *A* support-founded model *of* Σ_x *is a support-founded interpretation which is a model of* Σ_x.

Then using these support-founded models, the following characterization of REBAF semantics is:

Proposition 2 *Let* $\mathbf{REBAF} = \langle \mathbf{A}, \mathbf{R}_a, \mathbf{R}_e, \mathbf{s}, \mathbf{t}, \mathbf{P} \rangle$. *Let* $U = (S, \Gamma, \Delta)$ *be a structure on* **REBAF**.

1. U *is admissible iff there exists* $\mathcal{I}$ *support-founded model of* $\Sigma_d(\mathbf{REBAF})$ *(in the sense of Definition 16) with* $S_{\mathcal{I}} = S$, $\Gamma_{\mathcal{I}} = \Gamma$ *and* $\Delta_{\mathcal{I}} = \Delta$.
2. U *is complete iff there exists* $\mathcal{I}$ *support-founded model of the union* $(\Sigma_d(\mathbf{REBAF}) \cup \Sigma_r(\mathbf{REBAF}))$ *(in the sense of Definition 16) with* $S_{\mathcal{I}} = S$, $\Gamma_{\mathcal{I}} = \Gamma$ *and* $\Delta_{\mathcal{I}} = \Delta$.
3. U *is a preferred structure iff there exists* $\mathcal{I}$ $\subseteq$*-maximal support-founded model of* $\Sigma_d(\mathbf{REBAF})$ *(in the sense of Definition 16) with* $S_{\mathcal{I}} = S$, $\Gamma_{\mathcal{I}} = \Gamma$ *and* $\Delta_{\mathcal{I}} = \Delta$.
4. U *is the grounded structure iff* $S = S_{\mathcal{I}}$, $\Gamma_{\mathcal{I}} = \Gamma$ *and* $\Delta_{\mathcal{I}} = \Delta$ *where* $\mathcal{I}$ *is a* $\subseteq$*-minimal support-founded model of* $(\Sigma_d(\mathbf{REBAF}) \cup \Sigma_r(\mathbf{REBAF}))$ *(in the sense of Definition 16).*
5. U *is stable iff there exists* $\mathcal{I}$ *support-founded model of* $\Sigma_s(\mathbf{REBAF})$ *(in the sense of Definition 16) with* $S_{\mathcal{I}} = S$, $\Gamma_{\mathcal{I}} = \Gamma$ *and* $\Delta_{\mathcal{I}} = \Delta$.

Using this proposition, the preferred semantics produce the following results on the previous examples.

Example 7 (cont'd): The $\subseteq$-maximal support-founded models $\mathcal{I}$ of $\Sigma_d(\mathbf{REBAF})$ (in the sense of Definition 16) correspond to the preferred structure $(\{b, c, e, f\}, \{\beta, \pi\}, \{\epsilon\})$. □

Example 8 (cont'd): The $\subseteq$-maximal support-founded models $\mathcal{I}$ of $\Sigma_d(\mathbf{REBAF})$ (in the sense of Definition 16) correspond to the preferred structure $(\{b, d\}, \{\beta, \pi\}, \{\epsilon\})$. □

Example 9 (cont'd): The $\subseteq$-maximal support-founded models $\mathcal{I}$ of $\Sigma_d(\mathbf{REBAF})$ (in the sense of Definition 16) correspond to the preferred structure $(\{a, c, d\}, \{\alpha, \beta, \pi\}, \{\epsilon\})$. We see here that the existence of an $\mathbf{IST}$ for e was not enough for having e in the structure since there is no model $\mathcal{I}$ satisfying $eAcc(b)$. □

Example 10 (cont'd): The $\subseteq$-maximal support-founded models $\mathcal{I}$ of $\Sigma_d(\mathbf{REBAF})$ (in the sense of Definition 16) correspond to the preferred structure $(\{b, d\}, \{\beta, \pi\}, \{\epsilon\})$. The models that satisfy $eAcc(e)$ are not kept since they are not support-founded (no $\mathbf{IST}$ for e that could be satisfied). □

Example 11 (cont'd): The $\subseteq$-maximal support-founded models $\mathcal{I}$ of $\Sigma_d(\mathbf{REBAF})$ (in the sense of Definition 16) correspond to the preferred structure $(\varnothing, \varnothing, \{\alpha_1, \alpha_2, \beta_1, \beta_2\})$. The models that satisfy $eAcc(a)$ (resp. $eAcc(b)$, $eAcc(c)$ or $eAcc(d)$) are not kept since they are not support-founded (no $\mathbf{IST}$ for these arguments that could be satisfied). □

Example 12 (cont'd): The $\subseteq$-maximal support-founded models $\mathcal{I}$ of $\Sigma_d(\mathbf{REBAF})$ (in the sense of Definition 16) correspond to the preferred structure $(\{a, b, c\}, \varnothing, \{\alpha, \gamma\})$. The models that satisfy $eAcc(d)$ (resp. $eAcc(e)$, $eVal(\beta)$) are not kept since they are not support-founded (no $\mathbf{IST}$ for these arguments or this interaction that could be satisfied). □

Example 14 (cont'd): The $\subseteq$-maximal support-founded models $\mathcal{I}$ of $\Sigma_d(\mathbf{REBAF})$ (in the sense of Definition 16) correspond to the preferred structure $(\mathbf{A}, \mathbf{R}_a, \mathbf{R}_e)$. □

5 Conclusion

In this work, we have solved a specific issue concerning the handling of collective interactions in the logical translation of argumentation frameworks with higher-order attacks and evidential supports (REBAF). The resulting encoding is quite similar to the existing one for REBAF without collective interactions but the impact is much more important when support cycles exist and implies several new definitions.

Thus the notion of directed cycle of supports (DCS) has been improved. A new notion has also introduced here, the impacting support trees ($\mathbf{IST}$), in order to improved the notion of support-founded models.

Then using all these elements, we provided a characterization of admissible (resp. complete, preferred, stable and grounded) structures in the presence of support cycles and collective interactions.

References

[1] R. Arisaka and K. Satoh. Voluntary Manslaughter? A Case Study with Meta-Argumentation with Supports. In *Proc. of JSAI-isAI 2016. LNCS, vol 10247*, pages 241–252. Springer, 2017.

[2] P. Baroni, F. Cerutti, M. Giacomin, and G. Guida. AFRA: Argumentation framework with recursive attacks. *Intl. Journal of Approximate Reasoning*, 52:19–37, 2011.

[3] H. Barringer, D.M. Gabbay, and J. Woods. Temporal dynamics of support and attack networks : From argumentation to zoology. In D. Hutter and W. Stephan, editors, *Mechanizing Mathematical Reasoning. LNAI 2605*, pages 59–98. Springer Verlag, 2005.

[4] G. Boella, D. M. Gabbay, L. van der Torre, and S. Villata. Support in abstract argumentation. In *Proc. of COMMA*, pages 111–122. IOS Press, 2010.

[5] G. Boella, L. W. N. van der Torre, and S. Villata. Attack relations among dynamic coalitions. In *Proc. of BNAIC*, pages 25–32. Universiteit Twente, Enschede, 2008.

[6] C. Cayrol, A. Cohen, and M-C. Lagasquie-Schiex. Towards a new framework for recursive interactions in abstract bipolar argumentation. In *Proc. of COMMA*, pages 191–198. IOS Press, 2016.

[7] C. Cayrol, J. Fandinno, L. Fariñas del Cerro, and M-C. Lagasquie-Schiex. Valid attacks in argumentation frameworks with recursive attacks. In *Proc. of Commonsense Reasoning*, volume 2052. CEUR Workshop Proceedings, 2017.

[8] C. Cayrol, J. Fandinno, L. Fariñas del Cerro, and M.-C. Lagasquie-Schiex. Argumentation frameworks with recursive attacks and evidence-based support. In *Proc. of FoIKS*, volume LNCS 10833, pages 150–169. Springer-Verlag, 2018.

[9] C. Cayrol and M-C. Lagasquie-Schiex. Gradual valuation for bipolar argumentation frameworks. In *Proc. of ECSQARU*, pages 366–377. Springer, 2005.

[10] C. Cayrol and M-C. Lagasquie-Schiex. Bipolarity in argumentation graphs: towards a better understanding. *Intl. J. of Approximate Reasoning*, 54(7):876–899, 2013.

[11] Claudette Cayrol and Marie-Christine Lagasquie-Schiex. Logical encoding of argumentation frameworks with higher-order attacks and evidential supports. *International Journal on Artificial Intelligence Tools*, 29(3-4):2060003:1–2060003:50, June 2020.

[12] A. Cohen, S. Gottifredi, A. J. García, and G. R. Simari. A survey of different approaches to support in argumentation systems. *The Knowledge Engineering Review*, 29:513–550, 2014.

[13] A. Cohen, S. Gottifredi, A. J. García, and G. R. Simari. An approach to abstract argumentation with recursive attack and support. *J. Applied Logic*, 13(4):509–533, 2015.

[14] A. Cohen, S. Gottifredi, A. J. García, and G. R. Simari. On the acceptability semantics of argumentation frameworks with recursive attack and support. In *Proc. of COMMA*, pages 231–242. IOS Press, 2016.

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

[16] Phan Minh Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77:321–357, 1995.

[17] D. M. Gabbay. Fibring argumentation frames. *Studia Logica*, 93:231–295, 2009.

[18] D. M. Gabbay. Semantics for higher level attacks in extended argumentation frames. *Studia Logica*, 93:357–381, 2009.

[19] N. Karacapilidis and D. Papadias. Computer supported argumentation and collaborative decision making: the HERMES system. *Information systems*, 26(4):259–277, 2001.

[20] Marie-Christine Lagasquie-Schiex. Handling support cycles in the logical encoding of argumentation frameworks with higher-order attacks and evidential supports. Rapport de recherche IRIT/RR- -2021- -04- -FR, IRIT, France, May 2021.

[21] Sanjay Modgil. Reasoning about preferences in argumentation frameworks. *Artificial Intelligence*, 173:901–934, 2009.

[22] F. Nouioua and V. Risch. Bipolar argumentation frameworks with specialized supports. In *Proc. of ICTAI*, pages 215–218. IEEE Computer Society, 2010.

[23] F. Nouioua and V. Risch. Argumentation frameworks with necessities. In *Proc. of SUM*, pages 163–176. Springer-Verlag, 2011.

[24] N. Oren and T. J. Norman. Semantics for evidence-based argumentation. In *Proc. of COMMA*, pages 276–284. IOS Press, 2008.

[25] N. Oren, C. Reed, and M. Luck. Moving between argumentation frameworks. In *Proc. of COMMA*, pages 379–390. IOS Press, 2010.

[26] I. Rahwan and G. Simari. *Argumentation in Artificial Intelligence*. Springer, 2009.

[27] B. Verheij. Deflog: on the logical interpretation of prima facie justified assumptions. *Journal of Logic in Computation*, 13:319–346, 2003.

[28] S. Villata, G. Boella, D. M. Gabbay, and L. van der Torre. Modelling defeasible and prioritized support in bipolar argumentation. *AMAI*, 66(1-4):163–197, 2012.

A Proofs

We recalled here a notation and some lemmas given in [11] that will be useful for our proof.

Notation 1 (Notation Appendix A.1 in [11]) *Let $U = (S, \Gamma, \Delta)$ be a structure of* REBAF*, and $x \in \mathbf{A} \cup \mathbf{R}_a \cup \mathbf{R}_e$. x will be said to be* defended *by U, iff every attack $\alpha \in \mathbf{R}_a$ with $\mathbf{t}(\alpha) = x$ is unactivable w.r.t. U. $Defended(U)$ will denote the set of elements that are defended by U.*
Note that $x \in Acc(U)$ iff $x \in Sup(U)$ and $x \in Defended(U)$.

Lemma 1 (Lemma 7 in [8] and Lemma Appendix A.1 in [11]) *Any conflict-free self-supporting structure U satisfies:*
$Acc(U) \subseteq \overline{UnAcc(U)} \subseteq \overline{Def(U)}$.

Lemma 2 (Lemma Appendix A.2 in [11]) *Any stable structure U satisfies: $\overline{Sup(U)} = UnSupp(U)$.*

The following lemme corresponds to Lemma Appendix A.3, in [11] but for collective interactions.

Lemma 3 *Let $U = (S, \Gamma, \Delta)$ be a structure and $x \notin \mathbf{P}$ be the target of a support y such that $y \in \Delta \cap Sup(U)$ and $s_y \subseteq S \cap Sup(U)$. Then, there exists a support z such that $t_z = x$, $z \in \Delta \cap Sup(U \setminus \{x\})$ and $s_z \subseteq S \cap Sup(U \setminus \{x\})$ and so $x \in Sup(U)$.*

Proof of Lemma 3
Assume that $x = t_y$ with $y \in \Delta \cap Sup(U)$ and $s_y \subseteq S \cap Sup(U)$. If $y \in Sup(U \setminus \{x\})$ and $s_y \subseteq Sup(U \setminus \{x\})$, the result is proved with $z = y$. In the other case, without loss of generality, it can be assumed that $s_y \not\subseteq Sup(U \setminus \{x\})$ (the reasoning with $y \notin Sup(U \setminus \{x\})$ would be similar). So $s_y \subseteq Sup(U) \setminus Sup(U \setminus \{x\})$. As $s_y \subseteq Sup(U)$, there is a tree of supported supports (*i.e.* the support and its source belong to $U \cap Sup(U)$) leading to an element of s_y and rooted in prima-facie elements. As $s_y \not\subseteq Sup(U \setminus \{x\})$, at least one support in this tree has x as its source. Then let us consider the shortest sub-tree of this tree that is rooted in prima-facie elements and ends with x. It follows that this subtree contains supported supports, is rooted in prima-facie elements and does not contain x. Taking z as the support targeting x in this subtree will end the proof.

Proof of Proposition 1.[17]

Let **REBAF** $= \langle \mathbf{A}, \mathbf{R}_a, \mathbf{R}_e, \mathbf{s}, \mathbf{t}, \mathbf{P} \rangle$.

1. (admissibility)

   $\Rightarrow$ Assume that the structure $U = (S, \Gamma, \Delta)$ is admissible. Let us define an interpretation $\mathcal{I}$ of Σ_d(**REBAF**). The idea is to define $\mathcal{I}$ by successively adding constraints that $\mathcal{I}$ should satisfy:
   - For all $x \in \mathbf{A} \cup \mathbf{R}_a \cup \mathbf{R}_e$,
     $\mathcal{I}(Arg(x)) =$ *true* iff $x \in \mathbf{A}$,
     $\mathcal{I}(Attack(x)) =$ *true* iff $x \in \mathbf{R}_a$ and
     $\mathcal{I}(ESupport(x)) =$ *true* iff $x \in \mathbf{R}_e$.
   - For all $x \in \mathbf{A} \cup \mathbf{R}_a \cup \mathbf{R}_e$, $\mathcal{I}(PrimaFacie(x)) =$ *true* iff $x \in \mathbf{P}$.
   - For all $x \in \mathbf{A} \cup \mathbf{R}_a \cup \mathbf{R}_e$, $\mathcal{I}(Supp(x)) =$ *true* iff $x \in Sup(U)$.
   - For all $x \in \mathbf{A} \cup \mathbf{R}_a \cup \mathbf{R}_e$, $\mathcal{I}(UnSupp(x)) =$ *true* iff $x \in UnSupp(U)$.
   - For all $x \in \mathbf{A}$, $\mathcal{I}(Acc(x)) =$ *true* iff $x \in S$ or ($x \notin S$, $x \notin Sup(U)$ and $x \in Defended(U)$).
   - For all $x \in \mathbf{A}$, $\mathcal{I}(NAcc(x)) =$ *true* iff $\mathcal{I}(Acc(x)) =$ *false*.
   - For all $x \in \mathbf{R}_a$ (resp. $\in \mathbf{R}_e$), $\mathcal{I}(Val(x)) =$ *true* iff $x \in \Gamma$ (resp. Δ) or ($x \notin \Gamma$ (resp. Δ), $x \notin Sup(U)$ and $x \in Defended(U)$).
   - For all $x \in \mathbf{A}$, $\mathcal{I}(eAcc(x)) =$ *true* iff
     ($\mathcal{I}(Acc(x)) =$ *true* and $\mathcal{I}(Supp(x)) =$ *true*).
   - For all $x \in \mathbf{R}_a \cup \mathbf{R}_e$, $\mathcal{I}(eVal(x)) =$ *true* iff
     ($\mathcal{I}(Val(x)) =$ *true* and $\mathcal{I}(Supp(x)) =$ *true*).

   Note that in the current case, $Sup(U)$ refers to Definition 5.
   We have to prove that $S_{\mathcal{I}} = S$, $\Gamma_{\mathcal{I}} = \Gamma$, $\Delta_{\mathcal{I}} = \Delta$, and that $\mathcal{I}$ is a support-founded model of Σ_d(**REBAF**). And for proving that $\mathcal{I}$ is a support-founded model of Σ_d(**REBAF**) it is sufficient to prove that $\mathcal{I}$ satisfies the formulae **(1)**, **(2)**, **(3)**, **(1bis)**, **(2bis)**, **(3bis)** and **(17)**, **(18)**, **(11)**, **(12)** and that is a support-founded interpretation.[18]

   ★ Let $x \in S_{\mathcal{I}}$. By definition of $S_{\mathcal{I}}$, $\mathcal{I}(eAcc(x)) =$ *true*, that is $\mathcal{I}(Acc(x)) =$ *true* and $\mathcal{I}(Supp(x)) =$ *true*. By definition of $\mathcal{I}(Acc)$ and $\mathcal{I}(Supp)$ it follows that $x \in S$. Conversely, given $x \in S$, it holds that $\mathcal{I}(Acc(x)) =$ *true*. As U is admissible, U is self-supporting, so $x \in Sup(U)$, then it holds that

[17] This proof is inspired by the proof of Proposition 6.1 in [11].
[18] By definition, formulae **(4)** to **(10)** are satisfied by $\mathcal{I}$.

$\mathcal{I}(Supp(x)) = \textit{true}$. As a consequence, $\mathcal{I}(eAcc(x)) = \textit{true}$ and $x \in S_{\mathcal{I}}$. Proving that $\Gamma_{\mathcal{I}} = \Gamma$ and $\Delta_{\mathcal{I}} = \Delta$ is similar.

★ Obviously $\mathcal{I}$ satisfies formulae **(3)**, **(2bis)**, **(3bis)**.

★ Let us first consider formula **(2)**. Let $y \in \mathbf{R}_a$ and $x \in \mathbf{A}$ with $x = t_y$, $\mathcal{I}(eVal(y)) = \textit{true}$ and $\mathcal{I}(eAcc(s_y)) = \textit{true}$ (so $\forall x_i \in \mathbf{s}(y)$, $\mathcal{I}(eAcc(x_i)) = \textit{true}$). Then $s_y \subseteq S$ and $y \in \Gamma$. Let us assume that $\mathcal{I}(NAcc(x)) = \textit{false}$. Then $\mathcal{I}(Acc(x)) = \textit{true}$, by definition of $\mathcal{I}(NAcc)$. As U is admissible, U is conflict-free, so x cannot belong to S, and, by definition of $\mathcal{I}(Acc)$, it follows that $x \in Defended(U)$ and so $y \in UnAct(U)$, that is y belongs to $UnAcc(U)$ or s_y is included in $UnAcc(U)$. However, y and all elements of s_y being elements of the admissible structure U, due to Lemma 1, we obtain a contradiction. Hence, we have proved that $\mathcal{I}(NAcc(x)) = \textit{true}$ and formula **(2)** is satisfied by $\mathcal{I}$. Proving that formula **(1)** is satisfied by $\mathcal{I}$ is similar.

★ Let us first consider formula **(17)**. Let x such that $\mathcal{I}(Supp(x)) = \textit{true}$. By definition of $\mathcal{I}(Supp)$, $x \in Sup(U)$. By definition of $Sup(U)$, either $x \in \mathbf{P}$ or x is the target of a support α such that $\alpha \in \Delta$, $\alpha \in Sup(U \setminus \{x\})$, $s_\alpha \subseteq S$ and $s_\alpha \subseteq Sup(U \setminus \{x\})$. In the first case, formula **(17)** is trivially satisfied by $\mathcal{I}$. In the second case, as $S = S_{\mathcal{I}}$ and $\Delta = \Delta_{\mathcal{I}}$ it holds that $\mathcal{I}(eAcc(s_\alpha)) = \textit{true}$ and $\mathcal{I}(eVal(\alpha)) = \textit{true}$. Hence formula **(17)** is satisfied by $\mathcal{I}$.

★ Let us consider formula **(1bis)**. In the case when $\mathcal{I}(PrimaFacie(x)) = \textit{true}$, $x \in \mathbf{P}$, so $x \in Sup(U)$, hence $\mathcal{I}(Supp(x)) = \textit{true}$ and formula **(1bis)** is satisfied. Let us consider the case when $x \notin \mathbf{P}$ and x is the target of a support y such that $\mathcal{I}(eAcc(s_y)) = \textit{true}$ and $\mathcal{I}(eVal(y)) = \textit{true}$. We have to prove that $\mathcal{I}(Supp(x)) = \textit{true}$. As $S_{\mathcal{I}} = S$ and $\Delta_{\mathcal{I}} = \Delta$ it holds that $y \in \Delta$ and $s_y \subseteq S$. Moreover, as U is admissible, U is self-supporting, so y belongs to $Sup(U)$ and s_y is included in $Sup(U)$. From Lemma 3, it follows that $x \in Sup(U)$ hence $\mathcal{I}(Supp(x)) = \textit{true}$. So formula **(1bis)** is satisfied by $\mathcal{I}$.

★ Let us now consider formula **(18)**. Consider x such that $\mathcal{I}(\underline{UnSupp}(x)) = \textit{true}$. By definition of $\underline{\mathcal{I}(UnSupp)}$, $x \in \underline{UnSupp}(U)$. And, since $UnSupp(U) = \overline{Sup(U')}$ (where $U' = (\overline{Def_{\mathbf{A}}(U)}, \mathbf{R}_a, \overline{Def_{\mathbf{R}_e}(U)})$), $x \in \underline{\overline{Sup(U')}}$. So $x \notin \mathbf{P}$ and using the contrapositive of Lemma 3, applied to the structure U', it follows that for each support leading to x, either the support or its source (at least one of its components) is defeated by U, or the support or its source (at least one of its components) is itself not supported by U', hence belongs to $UnSupp(U)$. So the "only if" part of formula **(18)** is satisfied by $\mathcal{I}$.
For the "if" part, let us consider x such that $x \notin \mathbf{P}$ and for each support leading to x, either the support or its source (at least one of its components) is defeated by $(S_{\mathcal{I}}, \Gamma_{\mathcal{I}}, \Delta_{\mathcal{I}}) = U$, or the support or its source (at least one of its components) belongs to $UnSupp(U) = \overline{Sup(U')}$. As $U' \setminus \{x\} \subseteq U'$, it holds that $\overline{Sup(U')} \subseteq \overline{Sup(U' \setminus \{x\})}$. Hence, from Definition 5, it holds that $x \notin Sup(U')$, that is $x \in UnSupp(U)$ and so $\mathcal{I}(UnSupp(x)) = \textit{true}$. So formula **(18)** is satisfied by $\mathcal{I}$.

★ Let us now consider formula **(11)**. Let $\alpha \in \mathbf{R}_a$ and $x \in \mathbf{A}$ such that $x = t_\alpha$ and $\mathcal{I}(Acc(x)) = \textit{true}$. By definition of $\mathcal{I}(Acc)$, either $x \in S$ or ($x \notin S$, $x \notin Sup(U)$ and $x \in Defended(U)$). As U is admissible, in both cases, it holds that $\alpha \in UnAct(U)$. Then the fact that $\mathcal{I}$ satisfies formula **(11)** follows directly from the definition of $UnAct(U)$, the definition of $\mathcal{I}(Unsupp)$ and the fact that for an argument (resp. an attack) x, $\mathcal{I}(eAcc(x)) = \textit{true}$ (resp. $\mathcal{I}(eVal(x)) = \textit{true}$) iff $x \in S$ (resp. $x \in \Gamma$).
Proving that formula **(12)** is satisfied by $\mathcal{I}$ is similar.

★ Finally, we have to prove that $\mathcal{I}$ is support-founded.
Condition 2 of Definition 13 is trivially satisfied.
Consider now Condition 1. Let $x \in \mathbf{A}$ such that x is non prima-facie and there exists a DCS $\mathbf{C}$ containing x. Assume that $\mathcal{I}(eAcc(x)) = \textit{true}$. So $x \in U$ and, since U is admissible (so self-supporting) and x non prima-facie, then there exists at least one tree of supported supports $(x_0, \ldots, x_n)$ leading to x with any x_i belonging to U. Moreover, for all x_i, we have $x_i \in Sup(U)$. So following the definition of $\mathcal{I}$ we have either $\mathcal{I}(eAcc(x_i)) = \textit{true}$ or $\mathcal{I}(eVal(x_i)) = \textit{true}$ depending of the nature of x_i (argument or support). Thus there exists an impacting support tree $(x_0, \ldots, x_n)$ for x that is satisfied by $\mathcal{I}$. So $\mathcal{I}$ is a support-founded model. The proof for $x \in \mathbf{R}_e$ is similar.

$\Leftarrow$ Let $\mathcal{I}$ be a support-founded model of $\Sigma_d(\mathbf{REBAF})$. We have to prove that the structure $U = (S_{\mathcal{I}}, \Gamma_{\mathcal{I}}, \Delta_{\mathcal{I}})$ is admissible.

★ Let prove that U is conflict-free w.r.t. **REBAF**. If it is not the case, there exist $x \in S_{\mathcal{I}} \cup \Gamma_{\mathcal{I}} \cup \Delta_{\mathcal{I}}$ and $y \in \Gamma_{\mathcal{I}}$, with $s_y \subseteq S_{\mathcal{I}}$ and $t_y = x$.
By definition, it holds that $\mathcal{I}(eAcc(s_y)) =$ *true* and $\mathcal{I}(eVal(y)) =$ *true*. Moreover, in the case when $x \in S_{\mathcal{I}}$, it holds that $\mathcal{I}(eAcc(x)) =$ *true* and so $\mathcal{I}(Acc(x)) =$ *true* (as $\mathcal{I}$ satisfies formula **(2bis)**). Then it holds that $\mathcal{I}(NAcc(x)) =$ *false* (as $\mathcal{I}$ satisfies formula **(3)**). As a consequence, formula **(2)** is falsified.
In the case when $x \in \Gamma_{\mathcal{I}} \cup \Delta_{\mathcal{I}}$, it holds that $\mathcal{I}(eVal(x)) =$ *true* and so $\mathcal{I}(Val(x)) =$ *true* (as $\mathcal{I}$ satisfies formula **(3bis)**). As a consequence, formula **(1)** is falsified.
In both cases, there is a contradiction with $\mathcal{I}$ being a model of $\Sigma($**REBAF**$)$.

★ Let us prove that U is self-supporting. Assume that $x \in S_{\mathcal{I}}$ (resp. $x \in \Gamma_{\mathcal{I}} \cup \Delta_{\mathcal{I}}$). By definition, it holds that $\mathcal{I}(eAcc(x)) =$ *true* (resp. $\mathcal{I}(eVal(x)) =$ *true*). As $\mathcal{I}$ satisfies formula **(2bis)**, $\mathcal{I}(Supp(x)) =$ *true*. As $\mathcal{I}$ satisfies formula **(17)**, it holds that either $x \in \mathbf{P}$ or x is the target of a support x_n of source x_{n-1} such that $x_n \in \Delta_{\mathcal{I}}$ and $x_{n-1} \in S_{\mathcal{I}}$. In the first case, it holds that $x \in Sup(U)$. In the other case, it holds that $\mathcal{I}(eAcc(x_{n-1})) =$ *true* and $\mathcal{I}(eVal(x_n)) =$ *true*, and formula **(17)** can still be used, thus enabling to build a tree of supports. As U is finite and $\mathcal{I}$ is support founded, this process will end with $x_1 \in \mathbf{P}$ and $x_0 = \mathbf{s}(x_1) \in \mathbf{P}$. And so it can still be proved that $x \in Sup(U)$ w.r.t. Definition 5. Hence U is self-supporting.

★ It remains to prove that, given x an element of the structure, if x is the target of an attack α, then α is unactivable w.r.t. U. Assume that $x \in S_{\mathcal{I}}$ is the target of an attack α. By definition, it holds that $\mathcal{I}(eAcc(x)) =$ *true*. It follows that $\mathcal{I}(Acc(x)) =$ *true*. As $\mathcal{I}$ satisfies formula **(11)**, it follows that either there exists an attack β targeting α (or an element of s_α) with $\beta \in \Delta_{\mathcal{I}}$ and $s_\beta \subseteq S_{\mathcal{I}}$, or $\mathcal{I}$ satisfies $UnSupp(\alpha)$ (or $\mathcal{I}$ satisfies $UnSupp(s_\alpha)$, *i.e.* at least for one element x_i of s_α we have $\mathcal{I}(UnSupp(x_i)) =$ *true*).

- In the first case, it holds that α (resp. an element of s_α) belongs to $Def(U)$.
- In the second case, we prove that α (resp. an element of s_α) belongs to $UnSupp(U)$. For that purpose, we must prove that for any element x, if $\mathcal{I}$ satisfies $UnSupp(x)$, then $x \in UnSupp(U)$, or equivalently, if $x \in Sup(U')$ then $\mathcal{I}$ does not satisfy $UnSupp(x)$. Let us consider $x \in Sup(U')$. There is a tree of supports leading to x, rooted in prima-facie elements such that each support (and its source) in the tree is not defeated by U. As $\mathcal{I}$ satisfies formula **(18)**, the contrapositive of the "only if" part of formula **(18)** can be used for proving that each supported element y in this tree is such that $\mathcal{I}(UnSupp(y)) =$ *false*. The proof starts with the prima-facie elements of the set, and goes on by induction. Thus it can be proved that $\mathcal{I}(UnSupp(x)) =$ *false*.

So, in both cases, α is unactivable w.r.t. U.
The same reasoning can be done for $x \in \Gamma_{\mathcal{I}} \cup \Delta_{\mathcal{I}}$ using formula **(12)**. Hence, we can prove that U is admissible.

2. (complete semantics)

⇒ Assume that the structure $U = (S, \Gamma, \Delta)$ is complete. Let us build an interpretation $\mathcal{I}$ of $\Sigma_d($**REBAF**$) \cup \Sigma_r($**REBAF**$)$:

- We keep the same interpretation as the one used in Item 1 of the current proof except for Acc, Val.
- For all $x \in \mathbf{A}$, $\mathcal{I}(Acc(x)) =$ *true* iff $x \in S$ or ($x \notin S$ and $x \in Defended(U)$).
- For all $x \in \mathbf{R}_a$ (resp. $\in \mathbf{R}_e$), $\mathcal{I}(Val(x)) =$ *true* if and only $x \in \Gamma$ (resp. Δ) or ($x \notin \Gamma$ (resp. Δ) and $x \in Defended(U)$).

We have to prove that $S_{\mathcal{I}} = S$, $\Gamma_{\mathcal{I}} = \Gamma$ and $\Delta_{\mathcal{I}} = \Delta$, and that $\mathcal{I}$ is a support-founded model of $\Sigma_d($**REBAF**$) \cup \Sigma_r($**REBAF**$)$.

★ Note that if U is complete, for all $x \in \mathbf{A} \cup \mathbf{R}_a \cup \mathbf{R}_e$, if $x \notin S$ and $x \in Defended(U)$ then $x \notin Sup(U)$. So the above constraint expressed for the definition of $\mathcal{I}(Acc)$ (resp. $\mathcal{I}(Val)$), $x \notin S$ and $x \in Defended(U)$, is stronger than the one used for defining a model of an admissible structure ($x \notin S$, $x \notin Sup(U)$ and $x \in Defended(U)$).

Due to the above remark and the proof of Item 1 of this proof, it holds that $\mathcal{I}$ satisfies $S_{\mathcal{I}} = S$, $\Gamma_{\mathcal{I}} = \Gamma$, $\Delta_{\mathcal{I}} = \Delta$, and that $\mathcal{I}$ is a model of $\Sigma_d(\textbf{REBAF})$.

★ Now let prove that $\mathcal{I}$ satisfies formulae **(13)** and **(14)**. Let us consider formula **(13)**. Let $x \in \mathbf{A}$ such that for each attack α targeting x, either $\mathcal{I}(UnSupp(\alpha)) = \textit{true}$, or $\mathcal{I}(UnSupp(s_\alpha)) = \textit{true}$, or α (or an element of s_α) is attacked by β with $\beta \in \Gamma_{\mathcal{I}}$ and $s_\beta \subseteq S_{\mathcal{I}}$. Due to the definition of $\mathcal{I}(UnSupp)$, for each attack α targeting x, either $\alpha \in UnSupp(U)$, or an element of $s_\alpha \in UnSupp(U)$, or α (or an element of s_α) belongs to $Def(U)$. In other words, for each attack α targeting x, $\alpha \in UnAct(U)$, so $x \in Defended(U)$. Now, by definition of $\mathcal{I}(Acc)$, it holds that $\mathcal{I}(Acc(x)) = \textit{true}$. We have proved that $\mathcal{I}$ satisfies formula **(13)**. Proving that $\mathcal{I}$ satisfies formula **(14)** is similar.
So $\mathcal{I}$ is a model of $\Sigma_d(\textbf{REBAF}) \cup \Sigma_r(\textbf{REBAF})$.

★ It remains to prove that $\mathcal{I}$ is support-founded. For that purpose, the proof written in Item 1 of the current proof can be used as U is self-supporting.

⇐ Let $\mathcal{I}$ be a support-founded model of $\Sigma_d(\textbf{REBAF}) \cup \Sigma_r(\textbf{REBAF})$. We have to prove that the structure $U = (S_{\mathcal{I}}, \Gamma_{\mathcal{I}}, \Delta_{\mathcal{I}})$ is complete. For that purpose, it is enough to prove that $Acc(U)$ is included in $S_{\mathcal{I}} \cup \Gamma_{\mathcal{I}} \cup \Delta_{\mathcal{I}}$. Consider $x \in \mathbf{A} \cap Acc(U)$. So $x \in Sup(U)$ and $x \in Defended(U)$. The first condition implies that $\mathcal{I}(Supp(x)) = \textit{true}$, as $\mathcal{I}$ satisfies formula **(1bis)** and following the definition of $Sup(U)$. The second condition means that for each attack α targeting x, either $\alpha \in UnSupp(U)$, or an element of $s_\alpha \in UnSupp(U)$, or α (or an element of s_α) belongs to $Def(U)$ (*i.e.* α –or an element of s_α– is attacked by $\beta \in U$ with $s_\beta \subseteq U$). So, since $\mathcal{I}$ is a support-founded models (so Condition 2 of the definition of a support-founded model holds) and the fact that if an element β belongs to (resp. s_β is included in) the structure then $\mathcal{I}(eVal(\beta))$ (resp. $\mathcal{I}(eAcc(s_\beta))$) is also *true*, the premisse of formula **(13)** is *true*, and as $\mathcal{I}$ satisfies formula **(13)**, it follows that $\mathcal{I}(Acc(x)) = \textit{true}$. As $\mathcal{I}$ satisfies formula **(2bis)** it holds that $\mathcal{I}(eAcc(x)) = \textit{true}$, so $x \in S_{\mathcal{I}}$. Similarly, it can be proved that for all $x \in \mathbf{R}_a \cap Acc(U)$ (resp. $x \in \mathbf{R}_e \cap Acc(U)$), $x \in \Gamma_{\mathcal{I}}$ (resp. $x \in \Delta_{\mathcal{I}}$). We have proved that U is a complete structure.
The proof is similar for any support or attack in $Acc(U)$.

3. (preferred semantics) Let $\mathcal{I}$ be an interpretation of a set of formulae Σ_x. Let $U_{\mathcal{I}}$ denote the structure $(S_{\mathcal{I}}, \Gamma_{\mathcal{I}}, \Delta_{\mathcal{I}})$. It is easy to see that $\mathcal{I}$ is a $\subseteq$-maximal support-founded model of Σ_x iff the structure $U_{\mathcal{I}}$ is $\subseteq$-maximal among all the structures of the form $U_{\mathcal{J}} = (S_{\mathcal{J}}, \Gamma_{\mathcal{J}}, \Delta_{\mathcal{J}})$, where $\mathcal{J}$ denotes a support-founded model of Σ_x. Then taking $\Sigma_x = \Sigma_d(\textbf{REBAF})$, it follows that the preferred structures correspond to the structures $U_{\mathcal{I}}$ where $\mathcal{I}$ is a $\subseteq$-maximal support-founded model of $\Sigma_d(\textbf{REBAF})$.

4. (grounded semantics) Let $\mathcal{I}$ be an interpretation of a set of formulae Σ_x. Let $U_{\mathcal{I}}$ denote the structure $(S_{\mathcal{I}}, \Gamma_{\mathcal{I}}, \Delta_{\mathcal{I}})$. It is easy to see that $\mathcal{I}$ is a $\subseteq$-minimal support-founded model of Σ_x iff the structure $U_{\mathcal{I}}$ is $\subseteq$-minimal among all the structures of the form $U_{\mathcal{J}}$, where $\mathcal{J}$ denotes a support-founded model of Σ_x. Taking $\Sigma_x = \Sigma_d(\textbf{REBAF}) \cup \Sigma_r(\textbf{REBAF})$, it follows that the grounded structure correspond to the structure $U_{\mathcal{I}}$ where $\mathcal{I}$ is a $\subseteq$-minimal support-founded model of $\Sigma_d(\textbf{REBAF}) \cup \Sigma_r(\textbf{REBAF})$.

5. (stable semantics)

   ⇒ Assume that the structure $U = (S, \Gamma, \Delta)$ is stable. Let us define an interpretation $\mathcal{I}$ of $\Sigma_s(\textbf{REBAF})$ as follows:
   - Once again, we keep the same interpretation as the one used in Item 1 of the current proof except for Acc, Val.
   - For all $x \in \mathbf{A}$, $\mathcal{I}(Acc(x)) = \textit{true}$ iff $x \in S$ or $x \notin Def(U)$.
   - For all $x \in \mathbf{R}_a$ (resp. $\in \mathbf{R}_e$), $\mathcal{I}(Val(x)) = \textit{true}$ iff $x \in \Gamma$ (resp. Δ) or $x \notin Def(U)$.

We have to prove that $S_{\mathcal{I}} = S$, $\Gamma_{\mathcal{I}} = \Gamma$ and $\Delta_{\mathcal{I}} = \Delta$, and that $\mathcal{I}$ is a support-founded model of $\Sigma_s(\textbf{REBAF})$. And, for proving that $\mathcal{I}$ is a support-founded model of $\Sigma_s(\textbf{REBAF})$ it is sufficient to prove that $\mathcal{I}$ satisfies formulae **(1)**, **(2)**, **(3)**, **(1bis)**, **(2bis)**, **(3bis)** and **(17)**, **(18)**, **(15)**, **(16)**, **(19)** and that $\mathcal{I}$ is support-founded.

★ Let $x \in S_{\mathcal{I}}$. By definition, $\mathcal{I}(Acc(x)) =$ *true* and $\mathcal{I}(Supp(x)) =$ *true*. By definition of $\mathcal{I}(Acc)$ and $\mathcal{I}(Supp)$, it follows that $x \in Sup(U)$ and ($x \in S$ or $x \notin Def(U)$). Following Lemma 2, $x \notin UnSupp(U)$ and ($x \in S$ or $x \notin Def(U)$). If $x \notin S$, as U is stable, it follows that $x \in Def(U)$ or $x \in UnSupp(U)$. We obtain a contradiction, hence $x \in S$.
Conversely, given $x \in S$, it holds that $\mathcal{I}(Acc(x)) =$ *true*. As U is stable, U is self-supporting, so $x \in Sup(U)$, then it holds that $\mathcal{I}(Supp(x)) =$ *true*. As a consequence, $\mathcal{I}(eAcc(x)) =$ *true* and $x \in S_{\mathcal{I}}$. Proving that $\Gamma_{\mathcal{I}} = \Gamma$ and $\Delta_{\mathcal{I}} = \Delta$ is similar.

★ Obviously $\mathcal{I}$ satisfies formulae **(3)**, **(2bis)**, **(3bis)**.

★ Let us first consider formula **(2)**. Let $y \in \mathbf{R}_a$ and $x \in \mathbf{A}$ with $x = t_y$, $\mathcal{I}(eVal(y) =$ *true* and $\mathcal{I}(eAcc(s_y) =$ *true*. Then $s_y \subseteq S$ and $y \in \Gamma$, and it holds that $x \in Def(U)$. As U is stable, U is conflict-free, so x cannot belong to S. Hence we have $x \notin S$ and $x \in Def(U)$, or equivalently $\mathcal{I}(Acc(x)) =$ *false*, by definition of $\mathcal{I}(Acc)$ and then $\mathcal{I}(NAcc(x)) =$ *true*, by definition of $\mathcal{I}(NAcc)$. We have proved that $\mathcal{I}$ satisfies formula **(2)**. Proving that formula **(1)** is satisfied by $\mathcal{I}$ is similar.

★ Proving that $\mathcal{I}$ satisfies formulae **(1bis)**, **(17)**, **(18)** can be done with exactly the same reasoning as the one used in Item 1 of the current proof.

★ Let us now consider formula **(15)**. Let $x \in \mathbf{A}$ such that $\mathcal{I}(Acc(x)) =$ *false*. By definition of $\mathcal{I}(Acc)$, it holds that $x \notin S$ and $x \in Def(U)$. So, there is $y \in \Gamma$ with $x = t_y$ and $s_y \subseteq S$. Hence, there is $y \in \Gamma_{\mathcal{I}}$ with $x = t_y$ and $s_y \subseteq S_{\mathcal{I}}$, or equivalently, there is $y \in \mathbf{R}_a$ with $x = t_y$ and $\mathcal{I}(eVal(y)) =$ *true* and $\mathcal{I}(eAcc(s_y)) =$ *true*. We have proved that $\mathcal{I}$ satisfies formula **(15)**. Proving that formula **(16)** is satisfied by $\mathcal{I}$ is similar.

★ Lastly, we consider formula **(19)**. Let $x \in \mathbf{A} \cup \mathbf{R}_a \cup \mathbf{R}_e$ such that $\mathcal{I}(Supp(x)) =$ *false*. By definition of $\mathcal{I}(Supp)$, $x \notin Sup(u)$. Due to Lemma 2, it follows that $x \in UnSupp(U)$, hence $\mathcal{I}(UnSupp(x)) =$ *true*, by definition of $\mathcal{I}(UnSupp)$. We have proved that $\mathcal{I}$ satisfies formula **(19)**. So $\mathcal{I}$ is a model of $\Sigma_s(\textbf{REBAF})$.

★ It remains to prove that $\mathcal{I}$ is support-founded. For that purpose, the proof written in Item 1 of the current proof can be used as U is self-supporting.

⇐ Let $\mathcal{I}$ be a support-founded model of $\Sigma_s(\textbf{REBAF})$. We have to prove that the structure $U = (S_{\mathcal{I}}, \Gamma_{\mathcal{I}}, \Delta_{\mathcal{I}})$ is stable.
As noted in Definition 7, it is sufficient to prove that U is conflict-free, self-supporting and satisfies $\overline{U} \subseteq UnAcc(U)$.
As $\Sigma_s(\textbf{REBAF})$ contains $\Sigma(\textbf{REBAF})$, from Proposition 6.1 in [11], we know that the structure U is conflict-free. Moreover, $\Sigma_s(\textbf{REBAF})$ contains formulae **(17)**, **(18)**. So, with exactly the same reasoning as the one used in Item 1 for the admissible case, it can be proved that U is self-supporting.
It remains to prove that $\overline{U} \subseteq UnAcc(U)$. Let $x \in \mathbf{A}$ such that $x \in \overline{U}$. So $x \notin S_{\mathcal{I}}$ and by definition of $S_{\mathcal{I}}$, $\mathcal{I}(eAcc(x)) =$ *false*. As $\mathcal{I}$ satisfies formula **(2bis)**, it follows that $\mathcal{I}(Acc(x)) =$ *false* or $\mathcal{I}(Supp(x)) =$ *false*. In the case when $\mathcal{I}(Acc(x)) =$ *false*, as $\mathcal{I}$ satisfies formula **(15)**, it follows that $x \in Def(U)$. If $\mathcal{I}(Acc(x)) =$ *true*, it holds that $\mathcal{I}(Supp(x)) =$ *false*. As $\mathcal{I}$ satisfies formula **(19)**, it follows that $\mathcal{I}(UnSupp(x)) =$ *true*, so $x \in UnSupp(U)$ (following Condition 2 of Definition 13 since $\mathcal{I}$ is support-founded). In both cases, we have that $x \in UnAcc(U)$. We have proved that U is stable.