Logical Encoding of Argumentation Frameworks with Higher-order Attacks and Evidential Supports

Claudette Cayrol,
M-Christine Lagasquie-Schiex

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

Tech. Report
IRIT/RR- -2019- -03- -FR

Avril 2019

Abstract

We propose 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. Our purpose is to separate the logical expression of the meaning of an attack or an evidential support (simple or higher-order) from the logical expression of acceptability semantics. We consider semantics which specify the conditions under which the arguments (resp. the attacks/supports) are considered as accepted, directly on the extended framework, without translating the original framework into a Dung's argumentation framework. We characterize the output of a given framework in logical terms (namely as particular models of a logical theory). Our proposal applies to the particular case of Dung's frameworks, enabling to recover standard extensions.

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 [1]. Formal abstract frameworks have greatly eased the modelling and study of argumentation. The original Dung's argumentation framework (AF) [2] 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 [3, 4]. In [5], 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 [6], necessary support [7, 8], evidential support [9, 10]). 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. [11] 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 [13] 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 [14]), or for modelling situations where an attack might be defeated by an argument, without contesting the acceptability of the source of the attack [15]. Attacks to attacks and supports have been first considered in [16] with higher level networks, then in [17]; and more generally, [18] 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 [19]).

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 argument 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 [9] that allows to distinguish between two different kinds of arguments: *prima-facie* and *standard arguments*. *Prima-facie* arguments were already present in [4] 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 are considered as prima-facie arguments while b is 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 [15, 18, 20], or by reducing higher-order attacks to first-order joint attacks [21]. More recently, alternative acceptability semantics have been defined in a direct way for argumentation frameworks with higher-order attacks [22] or for higher-order attacks and supports (necessary supports: [23], evidential supports: [24]). 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.

In this paper, we propose a logical encoding of argumentation frameworks with higher-order attacks and evidential supports. Our purpose is (1) to characterize in a logical way the meaning of an attack or an evidential support (simple or higher-order) (2) to encode the acceptance conditions for arguments, attacks and supports proposed in [24] and (3) to characterize the outputs of the framework in logical terms, thus enabling to use logical tools for computational issues.

The connection between abstract argumentation and logics goes back to the seminal work of Dung, where a translation from an AF to a logic program was given. This line of research has been pursued with other kinds of translation *e.g.* in [25]. In [26], an AF with first-order joint attacks is encoded in a propositional logic augmented with strong negation. Other works have encoded acceptance conditions as logical formulae of a first-order theory *e.g.* [27], or defined a logical language for expressing the dynamics of a framework *e.g.* [28]. To the best of our knowledge, these works do not consider higher-order attacks.

The paper is organized as follows: the necessary background is given in Section 2; the logical encoding for frameworks with only higher-order attacks (RAF) is presented in sections 3 and 4 and the logical encoding for frameworks with higher-order attacks and evidential supports (REBAF) is presented in sections 5 and 6; some related works are discussed in Section 7, and Section 8 concludes the paper. The proofs are given in Appendix A.

Note that this paper is an extended version of [29] in which only the case of RAF was taken into account. First, we extend this previous work on RAF with examples and proofs, then we consider the case of REBAF using our RAF proposals as basic components. As REBAF is a much more complex formalism than RAF, starting with RAF encoding makes the work reported easier to read and understand.

2 Background

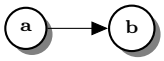
2.1 The Standard Abstract Framework

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

Definition 1 *[30] 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 *[30] 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 Attacks

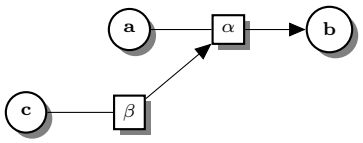
We consider a framework that allows representing both simple and higher-order attacks, *i.e.* attacks from an argument to either another argument or another attack. Such a framework has been usually called "recursive argumentation framework" in literature. So we keep this latter expression, even it is not completely satisfactory.

Definition 3 *[22] A* recursive argumentation framework (RAF) *is a tuple* $\langle \mathbf{A}, \mathbf{R}, \mathbf{s}, \mathbf{t} \rangle$ *where* $\mathbf{A}$ *is a finite and non-empty set of arguments,* $\mathbf{R}$ *is a finite set disjunct from* $\mathbf{A}$ *representing attack names,* $\mathbf{s}$ *is a function from* $\mathbf{R}$ *to* $\mathbf{A}$ *mapping each interaction to its source, and* $\mathbf{t}$ *is a function from* $\mathbf{R}$ *to* $(\mathbf{A} \cup \mathbf{R})$ *mapping each interaction to its target.*

Note that an AF can be viewed as a particular RAF with $\mathbf{t}$ being a mapping from $\mathbf{R}$ to $\mathbf{A}$.

A RAF can also be graphically represented.

Example 4 *An attack named* α *(with* $\mathbf{s}(\alpha) = a$ *and* $\mathbf{t}(\alpha) = b \in \mathbf{A}$*) being the target of an attack* β *with* $\mathbf{s}(\beta) = c$ *is represented by:*



(arguments are in a circle and attack names are in a square)

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

□

Acceptability semantics for argumentation frameworks with higher-order attacks have been defined in a direct way in [22]. The idea is to specify the conditions under which the arguments are considered as accepted directly on the extended framework, without translating the original framework into an AF. Moreover, due to the defeasible nature of attacks (attacks may be affected by other attacks), conditions under which the attacks are accepted must also be specified. Indeed, some attacks may not be "valid", in the sense that they cannot defeat the argument or attack they are targeting. So, acceptability conditions for arguments should be given with respect to valid attacks and conversely attacks should be declared valid with respect to other arguments or attacks. For instance, the fact that two arguments may be conflicting depends on the validity of the attack between them. Hence, the notion of extension (set of arguments) is replaced by a pair of a set of arguments and a set of attacks, called a "structure".

Definition 4 *[22] Consider* $\mathbf{RAF} = \langle \mathbf{A}, \mathbf{R}, \mathbf{s}, \mathbf{t} \rangle$. *A* structure of **RAF** *is a pair* (S, Γ) *with* $S \subseteq \mathbf{A}$ *and* $\Gamma \subseteq \mathbf{R}$.

Intuitively, given a structure $U = (S, \Gamma)$, S contains the arguments that are accepted "owing to" U and Γ contains the attacks which are valid "owing to" U (the meaning of "owing to" depending on the considered semantics).

In the following, we recall the acceptability conditions for structures, and the definitions of the semantics that are given in [22]. The key notion is the fact that a set of arguments (resp. attacks) can be "defeated" (resp. "inhibited") wrt a given structure.

Definition 5 *[22] Consider* $\mathbf{RAF} = \langle \mathbf{A}, \mathbf{R}, \mathbf{s}, \mathbf{t} \rangle$. *Given* $U = (S, \Gamma)$ *a structure of* **RAF**. *Let* $a \in \mathbf{A}$ *and* $\alpha \in \mathbf{R}$.

- a *is* defeated w.r.t. U *iff there is* $\beta \in \Gamma$ *with* $\mathbf{s}(\beta) \in S$ *and* $\mathbf{t}(\beta) = a$,
- α *is* inhibited w.r.t. U *iff there is* $\beta \in \Gamma$ *with* $\mathbf{s}(\beta) \in S$ *and* $\mathbf{t}(\beta) = \alpha$.

$Def(U)$ (resp. $Inh(U)$) denotes the set of arguments (resp. attacks) that are defeated (resp. inhibited) w.r.t. U.

2.2.1 Conflict-free structures

The minimal requirement for a structure (S, Γ) is that two arguments of S cannot be related by an attack of the structure, and similarly there cannot be an attack grounded in S and whose target is an element of Γ. Formally:

Definition 6 *[22] Consider* $\mathbf{RAF} = \langle \mathbf{A}, \mathbf{R}, \mathbf{s}, \mathbf{t} \rangle$. *A structure* $U = (S, \Gamma)$ *of* **RAF** *is* conflict-free *iff* $S \cap Def(U) = \varnothing$ *and* $\Gamma \cap Inh(U) = \varnothing$.

2.2.2 Admissible structures

Acceptability (for an argument or an attack) is also relative to a structure. Intuitively, an argument (resp. an attack) is acceptable if every attack against it can be considered as "non-valid" because either the attack is inhibited or its source is defeated.

Definition 7 *[22] Consider* $\mathbf{RAF} = \langle \mathbf{A}, \mathbf{R}, \mathbf{s}, \mathbf{t} \rangle$. *Given a structure* $U = (S, \Gamma)$ *of* **RAF**. *Let* $a \in \mathbf{A}$ *and* $\alpha \in \mathbf{R}$.

- a *(resp.* α*) is* acceptable w.r.t. U *iff for each* $\beta \in \mathbf{R}$ *with* $\mathbf{t}(\beta) = a$ *(resp.* $\mathbf{t}(\beta) = \alpha$*), either* $\beta \in Inh(U)$ *or* $\mathbf{s}(\beta) \in Def(U)$.
- U *is* admissible *iff it is conflict-free and for each* $x \in (S \cup \Gamma)$, x *is acceptable w.r.t.* U.

$Acc(U)$ *denotes the set containing all acceptable arguments and attacks w.r.t.* U.

Remark: Let $\alpha \in \mathbf{R}$ with $\mathbf{t}(\alpha) = b \in \mathbf{A}$ (resp. $\mathbf{t}(\alpha) = \beta \in \mathbf{R}$). If α and $\mathbf{s}(\alpha)$ are unattacked, there is no admissible structure $U = (S, \Gamma)$ such that $b \in S$ (resp. $\beta \in \Gamma$).

Example 4 (cont'd): As a, c, β are not attacked, they are acceptable w.r.t. any structure. As c, β are not attacked, for any structure U, α cannot be acceptable w.r.t. U. Moreover, b is acceptable w.r.t. a structure U implies that α is defeated w.r.t. U, and then U contains β and c.
As a consequence, the admissible structures containing b are $(\{b, c\}, \{\beta\})$ and $(\{a, b, c\}, \{\beta\})$.

□

2.2.3 Complete, stable, preferred and grounded structures

For any pair of structures $U = (S, \Gamma)$ and $U' = (S', \Gamma')$, $U \subseteq U'$ means that $(S \cup \Gamma) \subseteq (S' \cup \Gamma')$. The structure U is $\subseteq$-maximal iff every structure U' that satisfies $U \subseteq U'$ also satisfies $U' \subseteq U$. Similarly, U is $\subseteq$-minimal iff every structure U' that satisfies $U' \subseteq U$ also satisfies $U \subseteq U'$.

Definition 8 *[22] Consider* $\mathbf{RAF} = \langle \mathbf{A}, \mathbf{R}, \mathbf{s}, \mathbf{t} \rangle$. *A structure* $U = (S, \Gamma)$ *of* **RAF** *is:*

- complete *iff it is conflict-free and* $Acc(U) = S \cup \Gamma$.
- stable *iff it is conflict-free and satisfies* $\mathbf{A} \setminus S \subseteq Def(U)$ *and* $\mathbf{R} \setminus \Gamma \subseteq Inh(U)$.
- preferred *iff it is a* $\subseteq$*-maximal admissible structure.*
- grounded *iff it is the* $\subseteq$*-minimal conflict-free structure* $U = (S, \Gamma)$ *satisfying* $Acc(U) \subseteq S \cup \Gamma$.[3]

It has been proved in [22] that usual properties of Dung's extensions also hold for structures:

[3]The definition for the grounded structure has been given in [31] that is an extended version of [22].

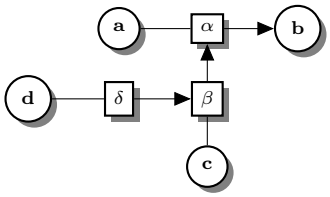
- A complete structure contains all the unattacked arguments and all the unattacked attacks.
- Every complete structure is admissible, every preferred structure is also complete and every stable structure is also preferred.

Moreover, it can be proved that the grounded structure is the $\subseteq$-minimal complete structure.[4]

Example 4 (cont'd): There is only one complete (resp. preferred) structure: $(\{a, b, c\}, \{\beta\})$.

□

Example 5 *The RAF depicted by the following figure is a variant of Ex. 4:*



In this case, for any structure U, β cannot be acceptable w.r.t. U. Moreover, α is acceptable w.r.t. U implies that U contains δ and d. As a consequence, there is only one complete (resp. preferred, stable, grounded structure): $(\{a, c, d\}, \{\alpha, \delta\})$.

□

2.2.4 D-structures

The notion of structure has been strengthened in order to obtain a conservative generalization of Dung's frameworks for the conflict-free, admissible, complete, stable and preferred semantics. It is worth to note that in an AF, each attack is considered as valid, in the sense that it may affect its target. The next definition strengthens the notion of structure by adding a condition on attacks that will force every acceptable attack to be valid.

Definition 9 *[22] Given* $\mathbf{RAF} = \langle \mathbf{A}, \mathbf{R}, \mathbf{s}, \mathbf{t} \rangle$.

1. *A* d-structure on $\mathbf{RAF}$ *is a structure* $U = (S, \Gamma)$ *such that* $(Acc(U) \cap \mathbf{R}) \subseteq \Gamma$.
2. *A* conflict-free (resp. admissible, complete, preferred, stable) d-structure *is a conflict-free (resp. admissible, complete, preferred, stable) structure which is also a d-structure.*

[4]The proof has been given in [31], the extended version of [22].

It follows from Def. 8 that every complete (resp. stable, preferred) structure of a RAF is a d-structure of this RAF. However it is not the case for admissible and conflict-free structures.
The conservative generalization proved in [22] relies upon a correspondence between a Dung's framework (and its extensions) and a "nonrecursive" RAF (and its d-structures), where a nonrecursive RAF is a RAF in which no attack targets another attack.

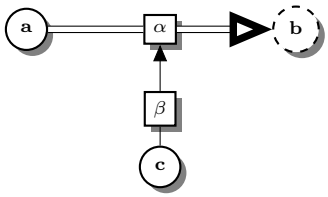
2.3 A Framework with Higher-Order Evidential Supports and Attacks

In this section, we recall the extension of [22] proposed in [24] for handling recursive attacks and evidence-based supports.

Definition 10 *[24] 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 (a solid border denotes prima-facie elements while a dashed border denotes standard elements; supports are represented by double edges):



□

Semantics of REBAF are defined in [24] using the extension of the notion of structure introduced in [22]. 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 $\mathbf{REBAF} = \langle \mathbf{A}, \mathbf{R}_a, \mathbf{R}_e, \mathbf{s}, \mathbf{t}, \mathbf{P} \rangle$.

Definition 11 *[24] 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[5] $\alpha \in \overline{\Gamma}$ 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 [22] in order to take into account the evidential supports.

Definition 12 *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:*[6]

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

Note that a standard element is supported if there is a "chain"[7] 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:

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

[5] 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. $\mathbf{R}_e$.

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

[7] Strictly speaking, it is not a chain, as each support may itself be the target of a support. However, we keep the word "chain" for simplicity.

- The set of *unactivable* attacks w.r.t. U is:
  $UnAct(U) \stackrel{\text{def}}{=} \{\alpha \in \mathbf{R}_a | \alpha \in UnAcc(U)$ or
  $\mathbf{s}(\alpha) \cap UnAcc(U) \neq \varnothing\}$

Definition 13 *[24] 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, $\alpha \in UnAct(U)$.*

$Acc(U)$ 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 14 *[24] A structure $U = (S, \Gamma, \Delta)$ is:*

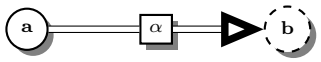
1. self-supporting *iff* $(S \cup \Gamma \cup \Delta) \subseteq Sup(U)$,
2. conflict-free *iff* $X \cap Def_Y(U) = \varnothing$ *for any* $(X, Y) \in \{(S, \mathbf{A}), (\Gamma, \mathbf{R}_a), (\Delta, \mathbf{R}_e)\}$,
3. admissible *iff it is conflict-free and* $S \cup \Gamma \cup \Delta \subseteq Acc(U)$,
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,*[8]
6. preferred *iff it is a $\subseteq$-maximal admissible structure,*
7. stable[9] *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 [24], which enable to prove for instance that there is a unique grounded structure.

Definitions 13 and 14 are illustrated on the following examples. Let us first consider a framework with no attack and only one support between two arguments. Different variants are considered depending on the set of prima-facie elements.

Example 6 *The support and its source are assumed to be prima-facie. The target is not prima-facie.*

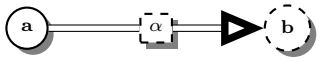


[8] The definition for the grounded extension is not given in [24] but can be easily proposed following the definition used in the AF case.

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

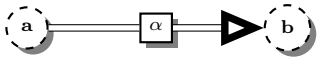
In this case, as α (resp. a) is prima-facie and not attacked, it is acceptable w.r.t. any structure. In contrast, b is not prime-facie, so b is supported w.r.t. a structure U implies that U contains the support α and its source a.
As a consequence, the structures $(\{a\}, \varnothing, \{\alpha\})$ and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

Example 7 *Only the source of the support is assumed to be prima-facie.*



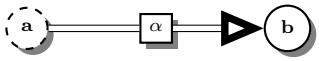
In this case, for any structure U, α is not supported w.r.t. U. It is the same for b. So the only admissible structures are $U = (\varnothing, \varnothing, \varnothing)$ and $U = (\{a\}, \varnothing, \varnothing)$. □

Example 8 *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 $U = (\varnothing, \varnothing, \{\alpha\})$. □

Example 9 *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, which is also a variant of Ex. 6, 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 [22] 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 [24]).

3 Logical Description of a RAF

First, we propose a logical description of a RAF, that allows an explicit representation of arguments, attacks and their properties (accepted argument, attacked argument, valid

attack, ...). We have been inspired by works in bioinformatics (see [32, 33]), where *metabolic networks* are used to describe the chemical reactions of cells; these reactions can be negative (inhibition of a protein) or positive (production of a new protein) and they can depend on other proteins or other reactions. A translation from metabolic networks to classical logic has been proposed in [33], which allows for the use of automated deduction methods for reasoning on these networks.

Given **RAF** a higher-order argumentation framework, $\Sigma(\mathbf{RAF})$ will denote the set of first-order logic formulae describing **RAF**.

3.1 Vocabulary

The following unary predicate symbols are used: Acc, $NAcc$, Val, $Attack$, Arg and the following unary functions symbols : T, S, with the following meaning:

- $Acc(x)$ (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
- $Attack(x)$ means "x is an attack"
- $Arg(x)$ means "x is an argument"
- $T(x)$ (resp. $S(x)$) denotes the target (resp. source) of x, when x denotes an attack

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. $\exists x \in Attack\ (\Phi(x))$)
- 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.

3.2 Logical theory

The formulae describing a given **RAF** can be partitioned in two sets Π and $\Pi(\mathbf{RAF})$:

- Π denotes the set of formulae describing the general behaviour of an attack, possibly recursive, in an argumentation framework, *i.e.* how an attack interacts with arguments and other attacks related to it.

- $\Pi(\mathbf{RAF})$ denotes the set of 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.

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

Using the vocabulary defined above, these constraints can be expressed by the following formulae:

(1) $\forall x \in Attack\, \forall y \in Attack$
$$\left(\begin{array}{l} (Val(y) \wedge (T(y) = x) \wedge Acc(S(y))) \\ \rightarrow \neg Val(x) \end{array} \right)$$

(2) $\forall x \in Arg\, \forall y \in Attack$
$$\left(\begin{array}{l} (Val(y) \wedge (T(y) = x) \wedge Acc(S(y))) \\ \rightarrow NAcc(x) \end{array} \right)$$

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

Two other formulae limit the domain to arguments or attacks.

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

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

The logical theory Π consists of the five above formulae.
Then, the logical encoding of specificities of a given *finite* RAF leads to the set $\Pi(\mathbf{RAF})$ consisting of the following formulae. Let $\mathbf{A} = \{a_1, \ldots a_n\}$ and $\mathbf{R} = \{\alpha_1, \ldots, \alpha_m\}$.

(6) $(S(\alpha) = a) \wedge (T(\alpha) = b)$ for all $\alpha \in \mathbf{R}$ 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) $\forall x\ (Attack(x) \leftrightarrow (x = \alpha_1) \vee \ldots \vee (x = \alpha_m))$

(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 $\alpha_i, \alpha_j \in \mathbf{R}$ with $i \neq j$

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

The logical theory $\Sigma(\mathbf{RAF})$ is the union of Π and $\Pi(\mathbf{RAF})$. It is obviously consistent.

Example 5 (cont'd): Using the equality axioms, a simplified version of $\Sigma(\mathbf{RAF})$ can be obtained (in particular tautologies are omitted):[10]

$\Sigma(\mathbf{RAF}) = \{(Val(\beta) \wedge Acc(c)) \rightarrow \neg Val(\alpha)$ (from (1)),
$(Val(\delta) \wedge Acc(d)) \rightarrow \neg Val(\beta)$ (from (1)),
$(Val(\alpha) \wedge Acc(a)) \rightarrow NAcc(b)$ (from (2)),
$NAcc(b) \rightarrow \neg Acc(b)$ (from (3)),
$NAcc(a) \rightarrow \neg Acc(a)$ (from (3)),
$NAcc(c) \rightarrow \neg Acc(c)$ (from (3)),
$NAcc(d) \rightarrow \neg Acc(d)$ (from (3))$\}$ □

In the particular case of a non-recursive RAF, formula (1) is a tautology. However, formula (2) cannot be simplified as it cannot be deduced that the attacks are valid. Indeed, the logical theory $\Sigma(\mathbf{RAF})$ only captures the description of **RAF** and is not concerned with the semantics of the framework (the logical description of the semantics is handled in the next section).

Example 3 (cont'd): Consider the non-recursive RAF containing only one attack from a to b.

$\Sigma(\mathbf{RAF})$ enables to deduce the formula $(Val(\alpha) \wedge Acc(a)) \rightarrow NAcc(b)$. Note that if $Val(\alpha)$ is assumed, we obtain $Acc(a) \rightarrow NAcc(b)$ and from $NAcc(b) \rightarrow \neg Acc(b)$ it follows that $Acc(a) \rightarrow \neg Acc(b)$ and so $Acc(b) \rightarrow \neg Acc(a)$. However, it cannot be deduced that $Acc(b) \rightarrow NAcc(a)$. Indeed, the predicate $NAcc$ allows for the representation of the direction of an attack between two arguments and avoids the contraposition of the attack. □

4 Logical Formalization of RAF Semantics

4.1 Logical Encoding of RAF Semantics

In presence of higher-order attacks, the conflict-freeness, defence, reinstatement and stability principles must take into account the fact that attacks might not be valid. Moreover, for each of these principles, two versions will be given, one for arguments and another one for attacks. Then, for each principle, we give a logical expression, thus leading to add formulae to the base $\Sigma(\mathbf{RAF})$ and producing new bases.

[10] The simplification will be applied for the other examples.

4.1.1 Conflict-freeness

The conflict-freeness principle is formulated as follows (for arguments and for attacks):

- If there is a valid attack between two arguments, they cannot be jointly accepted.
- If there is an attack from an accepted argument to an attack, these attacks cannot be both valid.

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

4.1.2 Defence

The idea is to claim that an argument a is defended by a set of arguments S if S weakens each attack α to a, either by attacking the source of α, or by attacking α itself. Moreover the defence should be obtained with valid attacks. So, the defence principle is formulated as follows (for arguments and for attacks):

- An attacked argument may be accepted only if for each attack to it, either the source or the attack itself is in turn attacked by a valid attack from an accepted argument.
- An attack may be valid only if for each attack to it, either the source or the attack itself is in turn attacked by a valid attack from an accepted argument.

These properties are expressed by the following formulae:[11]

(11) $\forall \alpha \in Attack$
$$\left(\begin{array}{l} Acc(t_\alpha) \\ \rightarrow \left(\begin{array}{l} \exists \beta \in Attack \\ (t_\beta \in \{s_\alpha, \alpha\} \wedge Val(\beta) \wedge Acc(s_\beta)) \end{array} \right) \end{array} \right)$$

(12) $\forall \alpha \in Attack\, \forall \delta \in Attack$
$$\left(\begin{array}{l} ((\delta = t_\alpha) \wedge Val(\delta)) \\ \rightarrow \left(\begin{array}{l} \exists \beta \in Attack \\ (t_\beta \in \{s_\alpha, \alpha\} \wedge Val(\beta) \wedge Acc(s_\beta)) \end{array} \right) \end{array} \right)$$

These formulae are added to the base $\Sigma(\mathbf{RAF})$, thus producing the base $\Sigma_d(\mathbf{RAF})$.

4.1.3 Reinstatement

Based on the previous notion of defence, the reinstatement principle is formulated as follows (for arguments and for attacks):

- An argument must be accepted provided that, for each attack to it, the source or the attack itself is in turn attacked by a valid attack from an accepted argument.

[11] Strictly speaking, $t_\beta \in \{s_\alpha, \alpha\}$ should be written as follows : $t_\beta = s_\alpha \vee t_\beta = \alpha$.

- An attack may be valid provided that, for each attack to it, either the source or the attack itself is in turn attacked by a valid attack from an accepted argument.

These properties are expressed by the following formulae:

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

(14) $\forall \delta \in Attack$
$$\left(\begin{array}{l} \left(\begin{array}{l} \forall \alpha \in Attack\ (t_\alpha = \delta \\ \rightarrow (\exists \beta \in Attack \\ \quad (t_\beta \in \{s_\alpha, \alpha\} \wedge Val(\beta) \wedge Acc(s_\beta)))) \end{array} \right) \\ \rightarrow Val(\delta) \end{array} \right)$$

These formulae are added to the base $\Sigma(\mathbf{RAF})$, thus producing the base $\Sigma_r(\mathbf{RAF})$.

4.1.4 Stability

The stability requirement can be formulated as follows (one for arguments and one for attacks):

- If an argument is not accepted, it must be attacked by a valid attack from an accepted argument.
- If an attack is not valid, it must be attacked by a valid attack from an accepted argument.

These properties are expressed by the following formulae:

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

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

These formulae are added to the base $\Sigma(\mathbf{RAF})$, thus producing the base $\Sigma_s(\mathbf{RAF})$.

Example 5 (cont'd): $\Sigma_d(\mathbf{RAF})$ is obtained from $\Sigma(\mathbf{RAF})$ by adding the formulae:
$Acc(b) \rightarrow (Val(\beta) \wedge Acc(c))$,
$\neg Val(\beta)$ and
$Val(\alpha) \rightarrow (Val(\delta) \wedge Acc(d))$.

$\Sigma_r(\mathbf{RAF})$ is obtained from $\Sigma(\mathbf{RAF})$ by adding the formulae:

$Acc(a)$,
$Acc(c)$,
$Acc(d)$,
$(Val(\beta) \wedge Acc(c)) \rightarrow Acc(b)$,
$Val(\delta)$ and
$(Val(\delta) \wedge Acc(d)) \rightarrow Val(\alpha)$. □

4.2 Characterizing Semantics of a RAF

We propose characterizations of the structures under different semantics in terms of models of the bases $\Sigma(\mathbf{RAF})$, $\Sigma_d(\mathbf{RAF})$, $\Sigma_r(\mathbf{RAF})$, $\Sigma_s(\mathbf{RAF})$.

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

- $S_\mathcal{I} = \{x \in \mathbf{A} | \mathcal{I}(Acc(x)) = \textit{true}\}$
- $\Gamma_\mathcal{I} = \{x \in \mathbf{R} | \mathcal{I}(Val(x)) = \textit{true}\}$

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

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

We have the following characterizations:

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

1. *U is conflict-free iff there exists $\mathcal{I}$ model of $\Sigma(\mathbf{RAF})$ with $S_\mathcal{I} = S$ and $\Gamma_\mathcal{I} = \Gamma$.*
2. *U is admissible iff there exists $\mathcal{I}$ model of $\Sigma_d(\mathbf{RAF})$ with $S = S_\mathcal{I}$ and $\Gamma_\mathcal{I} = \Gamma$.*
3. *U is complete iff there exists $\mathcal{I}$ model of $\Sigma_d(\mathbf{RAF}) \cup \Sigma_r(\mathbf{RAF})$ with $S = S_\mathcal{I}$ and $\Gamma_\mathcal{I} = \Gamma$.*
4. *U is a stable structure iff there exists $\mathcal{I}$ model of $\Sigma_s(\mathbf{RAF})$ with $S_\mathcal{I} = S$ and $\Gamma_\mathcal{I} = \Gamma$.*
5. *U is a preferred structure iff there exists $\mathcal{I}$ $\subseteq$-maximal model of $\Sigma_d(\mathbf{RAF})$ with $S_\mathcal{I} = S$ and $\Gamma_\mathcal{I} = \Gamma$.*
6. *U is the grounded structure iff $S = S_\mathcal{I}$ and $\Gamma_\mathcal{I} = \Gamma$ where $\mathcal{I}$ is a $\subseteq$-minimal model of $\Sigma_r(\mathbf{RAF})$.*[12]

[12] It also holds that U is the grounded structure iff $S = S_\mathcal{I}$ and $\Gamma_\mathcal{I} = \Gamma$ where $\mathcal{I}$ is a $\subseteq$-minimal model of $\Sigma_d(\mathbf{RAF}) \cup \Sigma_r(\mathbf{RAF})$. Considering $\Sigma_d(\mathbf{RAF}) \cup \Sigma_r(\mathbf{RAF})$ instead of $\Sigma_r(\mathbf{RAF})$ might be useful from a computational point of view, when searching for minimal models.

Example 5 (cont'd): There is only one complete structure: $(\{a, c, d\}, \{\alpha, \delta\})$. Indeed, every model $\mathcal{I}$ of $\Sigma_d(\mathbf{RAF}) \cup \Sigma_r(\mathbf{RAF})$ is such that $S_\mathcal{I} = \{a, c, d\}$ and $\Gamma_\mathcal{I} = \{\alpha, \delta\}$, in other words, every model of $\Sigma_d(\mathbf{RAF}) \cup \Sigma_r(\mathbf{RAF})$ satisfies $Acc(a)$, $Acc(c)$, $Acc(d)$, $Val(\delta)$, $Val(\alpha)$ and falsifies $Acc(b)$, $Val(\beta)$. An example of admissible (but not complete) structure is $(\{a, d\}, \{\delta\})$. Indeed, there is a model $\mathcal{I}$ of $\Sigma_d(\mathbf{RAF})$ with $S_\mathcal{I} = \{a, d\}$ and $\Gamma_\mathcal{I} = \{\delta\}$. □

D-structures can also be characterized. Let us recall that d-structures are particular structures in which acceptable attacks are forced to be valid. So, we consider the base $\Sigma(\mathbf{RAF})$ augmented with the formula that expresses the reinstatement principle for attacks, that is formula (14).

As said before, complete structures are d-structures. So we just have to complete Proposition 1 with the characterizations of conflict-free and admissible d-structures.

Proposition 2 *Let* $\mathbf{RAF} = \langle \mathbf{A}, \mathbf{R}, \mathbf{s}, \mathbf{t} \rangle$. *Let* $U = (S, \Gamma)$ *a structure on* **RAF**.

1. *U is a conflict-free d-structure iff there exists $\mathcal{I}$ model of $\Sigma(\mathbf{RAF}) \cup \{(14)\}$ with $S_\mathcal{I} = S$ and $\Gamma_\mathcal{I} = \Gamma$.*
2. *U is an admissible d-structure iff there exists $\mathcal{I}$ model of $\Sigma_d(\mathbf{RAF}) \cup \{(14)\}$ with $S = S_\mathcal{I}$ and $\Gamma_\mathcal{I} = \Gamma$.*

Example 5 (cont'd): From (14), the following formulae are obtained: $Val(\delta)$ and $(Val(\delta) \wedge Acc(d)) \rightarrow Val(\alpha)$.
An example of admissible (but not complete) d-structure is $(\{a, d\}, \{\alpha, \delta\})$. Indeed, there is a model $\mathcal{I}$ of $\Sigma_d(\mathbf{RAF}) \cup \{(14)\}$ with $S_\mathcal{I} = \{a, d\}$ and $\Gamma_\mathcal{I} = \{\alpha, \delta\}$. Note that $(\{a, d\}, \{\delta\})$ is an admissible structure but not an admissible d-structure. □

4.3 Case of AF

As said before, an AF can be viewed as a particular RAF. So we can consider the associated logical theory, which we denote by $\Sigma(\mathbf{AF})$ for simplicity. Moreover, in the particular case of an AF, the semantics recalled in Section2.1 assume that each attack is valid. As a consequence, the logical theory Π can be replaced by a logically equivalent theory built as follows: For each $(a, b) \in \mathbf{R}$, the attack from a to b is described by the formulae $Acc(a) \rightarrow NAcc(b)$ and $NAcc(b) \rightarrow \neg Acc(b)$.[13]

Then, the standard defence, reinstatement and stability principles are encoded with simplified versions of formulae (11), (13) and (15) (as attacks are never attacked, formulae (12), (14) and (16) would be tautologies). Let $\mathbf{AF} = \langle \mathbf{A}, \mathbf{R} \rangle$. For $x \in \mathbf{A}$, let $\mathbf{R}^-(x)$ denote the set of its attackers. For each principle, a set of formulae is provided, one for each argument:

- *Defence:* For each $x \in \mathbf{A}$,
$$Acc(x) \rightarrow (\wedge_{y \in \mathbf{R}^-(x)} (\vee_{z \in \mathbf{R}^-(y)} Acc(z)))$$

[13] This is a simplified version of formulae (2) and (3); moreover formula (1) being a tautology in the case of AF, it does not appear in $\Sigma(\mathbf{AF})$.

- *Reinstatement:* For each $x \in \mathbf{A}$,
$$(\wedge_{y \in \mathbf{R}^-(x)}(\vee_{z \in \mathbf{R}^-(y)} Acc(z))) \rightarrow Acc(x)$$

- *Stability:* For each $x \in \mathbf{A}$,
$$\neg Acc(x) \rightarrow (\vee_{y \in \mathbf{R}^-(x)} Acc(y))$$

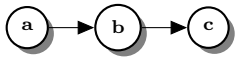
We denote by $\Sigma_d(\mathbf{AF})$ (resp. $\Sigma_r(\mathbf{AF})$, $\Sigma_s(\mathbf{AF})$) the logical theories obtained by adding all the formulae encoding defence (resp. reinstatement, stability) to $\Sigma(\mathbf{AF})$. Given $\mathcal{I}$ be an interpretation of $\Sigma(\mathbf{AF})$, we still denote by $S_\mathcal{I}$ the set $\{x \in \mathbf{A} \mid \mathcal{I}(Acc(x)) = \textit{true}\}$. If $\mathcal{I}$ is a model of $\Sigma(\mathbf{AF})$, $\mathcal{I}$ is said to be a $\subseteq$-maximal (resp. minimal) model of $\Sigma(\mathbf{AF})$ iff there is no model $\mathcal{I}'$ of $\Sigma(\mathbf{AF})$ such that $S_\mathcal{I} \subset S_{\mathcal{I}'}$ (resp. $S_{\mathcal{I}'} \subset S_\mathcal{I}$).

Then, the following characterizations can be obtained as a direct consequence of Proposition 1:

Consequence 1 *Let* $\mathbf{AF} = \langle \mathbf{A}, \mathbf{R} \rangle$. *Let* $S \subseteq \mathbf{A}$.

1. *S is conflict-free in $\langle \mathbf{A}, \mathbf{R} \rangle$ iff there exists $\mathcal{I}$ model of $\Sigma(\mathbf{AF})$ with $S_\mathcal{I} = S$.*
2. *S is admissible in $\langle \mathbf{A}, \mathbf{R} \rangle$ iff there exists $\mathcal{I}$ model of $\Sigma_d(\mathbf{AF})$ with $S = S_\mathcal{I}$.*
3. *S is a complete extension of $\langle \mathbf{A}, \mathbf{R} \rangle$ iff there exists $\mathcal{I}$ model of $\Sigma_d(\mathbf{AF}) \cup \Sigma_r(\mathbf{AF})$ with $S = S_\mathcal{I}$.*
4. *S is a stable extension of $\langle \mathbf{A}, \mathbf{R} \rangle$ iff there exists $\mathcal{I}$ model of $\Sigma_s(\mathbf{AF})$ with $S_\mathcal{I} = S$.*
5. *S is a preferred extension of $\langle \mathbf{A}, \mathbf{R} \rangle$ iff there exists $\mathcal{I}$ $\subseteq$-maximal model of $\Sigma_d(\mathbf{AF})$ with $S_\mathcal{I} = S$.*
6. *S is the grounded extension iff $S = S_\mathcal{I}$ where $\mathcal{I}$ is a $\subseteq$-minimal model of $\Sigma_r(\mathbf{AF})$.*

Example 10 *Consider the AF represented by:*



It can be encoded by the following simplified bases:
$\Sigma(\mathbf{AF}) = \{Acc(a) \rightarrow NAcc(b),$
$NAcc(b) \rightarrow \neg Acc(b),$
$Acc(b) \rightarrow NAcc(c),$
$NAcc(c) \rightarrow \neg Acc(c),$
$NAcc(a) \rightarrow \neg Acc(a)\}$
and $\Sigma_d(\mathbf{AF}) = \Sigma(\mathbf{AF}) \cup$
$\{\neg Acc(b),$
$Acc(c) \rightarrow Acc(a)\}.$
Every $\subseteq$-maximal model of $\Sigma_d(\mathbf{AF})$ satisfies $Acc(a)$, $Acc(c)$ and falsifies $Acc(b)$. That corresponds to the unique preferred extension $\{a, c\}$. □

5 Logical Description of a REBAF

In the second part of this paper, we propose a logical description of a REBAF, that allows an explicit representation of arguments, attacks, *evidential supports* and their properties. In order to use the same methodology as the one used for RAF, we consider a variant of REBAF 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. As a consequence, the definitions of $Def_X(U)$ and $Sup(U)$ given in Definition 12 can be simplified as follows:

Definition 15 *Given a structure* $U = (S, \Gamma, \Delta)$*,*

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

Given **REBAF** a higher-order argumentation framework, $\Sigma(\mathbf{REBAF})$ will denote the set of first-order logic formulae describing **REBAF**.

5.1 Vocabulary

We keep the vocabulary used for RAF (the unary predicate symbols Arg, $Attack$, Acc, $NAcc$, Val, and the unary functions symbols : T, S) as RAF is a particular case of REBAF. We add the unary predicate symbols $ESupport$ for denoting evidential supports, and $PrimaFacie$ for denoting prima-facie elements. Then we need symbols for denoting acceptability of elements. Let us recall that our purpose is 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 13, 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 we introduce the unary predicate symbols $Supp$ 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".

The previous remarks about the equality predicate, the quantifiers and the meaning of $NAcc(x)$ still hold.

5.2 Logical theory

As for the case of RAF, the formulae describing a given **REBAF** can be partitioned in two sets:

- The first set, still 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 $\Pi(\mathbf{REBAF})$, contains the formulae encoding the specificities of the current framework.

The meaning of an attack is similar as in the RAF case except that supports must be taken into account:[14]

- 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,[15] these constraints can be expressed 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) $\forall x \in (Attack \cup ESupport)$
$$((Val(x) \wedge Supp(x)) \leftrightarrow eVal(x))$$

[14] The main difference with the RAF case is the use of "e-accepted" (resp. "e-valid") in place of "accepted" (resp. "valid").

[15] We recall that s_α (resp. t_α) is short for $S(\alpha)$ (resp. $T(\alpha)$).

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

(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 $\mathbf{P} = \{x_1, \ldots x_l\}$.
[16]

(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 $i \neq j$

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 $\Sigma(\textbf{REBAF})$ will be given.[17]

Example 6 (cont'd):
$\Sigma(\textbf{REBAF}) = \{Supp(a)$ (from (1bis), (8ter)),
$Supp(\alpha)$ (from (1bis),(8ter)),
$(eAcc(a) \wedge eVal(\alpha)) \rightarrow Supp(b)$ (from (1bis)),

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

$(Supp(a) \wedge Acc(a)) \leftrightarrow eAcc(a)$ (from (2bis)),
$(Supp(b) \wedge Acc(b)) \leftrightarrow eAcc(b)$ (from (2bis)),
$(Supp(\alpha) \wedge Val(\alpha)) \leftrightarrow eVal(\alpha)$ (from (3bis))$\}$

□

Example 7 (cont'd):
$\Sigma(\mathbf{REBAF}) = \{Supp(a)$ (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): Let us recall that this is a variant of Ex. 6 where the attack β targeting α has been added.
$\Sigma(\mathbf{REBAF}) = \{(eVal(\beta) \wedge eAcc(c)) \rightarrow \neg Val(\alpha)$ (from (1)),
$Supp(a)$ (from (1bis), (8ter)),
$Supp(c)$ (from (1bis), (8ter)),
$Supp(\alpha)$ (from (1bis),(8ter)),
$Supp(\beta)$ (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))$\}$

□

It is worth noting that a RAF can be considered as a particular case of REBAF with an empty relation $\mathbf{R}_e$ and such that any argument or attack is prima-facie. In that particular case, it is easy to prove that the logical base $\Sigma(\mathbf{REBAF})$ is equivalent to the base obtained for RAF in Section 3.

6 Logical Formalization of REBAF semantics

6.1 Logical Encoding 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. So, the formulae encoding these principles are more complex than in RAF case.

6.1.1 Conflict-freeness

The conflict-freeness principle is 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.

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

6.1.2 Self-supporting

The self-supporting principle states that each supported element must receive evidential support. It can be 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)

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

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

6.1.3 Defence

As stated in Definition 13, 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, the principle corresponding to the previous item (ii) can be expressed by the following formulae that will be 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 $\Sigma_d(\mathbf{REBAF})$.

6.1.4 Reinstatement

The reinstatement principle can be expressed by the following formulae that will 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)

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

Formulae (13) and (14) are added to the base Σ_{ss}(**REBAF**), thus producing the base Σ_r(**REBAF**).

6.1.5 Stability

The stability principle can be expressed by the three following formulae that will be associated with formulae (17) and (18):[18]

(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 Σ_{ss}(**REBAF**), thus producing the base Σ_s(**REBAF**).

Example 6 (cont'd): Σ_{ss}(**REBAF**) is obtained from Σ(**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, Σ_d(**REBAF**) contains nothing more than Σ_{ss}(**REBAF**).

And finally Σ_r(**REBAF**) is obtained from Σ_{ss}(**REBAF**) by adding the formulae: $Acc(a)$, $Acc(b)$ and $Val(\alpha)$. □

Example 7 (cont'd): Σ_{ss}(**REBAF**) is obtained from Σ(**REBAF**) by adding the following formulae:

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

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

Once again, Σ_d(**REBAF**) contains nothing more than Σ_{ss}(**REBAF**).

And Σ_r(**REBAF**) is obtained from Σ_{ss}(**REBAF**) by adding the formulae: $Acc(a)$, $Acc(b)$ and $Val(\alpha)$. □

Example 2 (cont'd): Σ_{ss}(**REBAF**) is obtained from Σ(**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 Σ_d(**REBAF**) is obtained from Σ_{ss}(**REBAF**) by adding formulae among which:

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

Σ_r(**REBAF**) is obtained from Σ_{ss}(**REBAF**) by adding the formulae:

$Acc(a)$
$Acc(b)$
$Acc(c)$
$Val(\beta)$
$(UnSupp(c) \vee UnSupp(\beta)) \rightarrow Val(\alpha)$

Σ_s(**REBAF**) is obtained from Σ_{ss}(**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 $x \in \{a, c, \alpha, \beta\}$ □

6.2 Characterizing Semantics of a REBAF

Similarly as for RAF case, we propose characterizations of the REBAF structures under different semantics in terms of models of the bases Σ(**REBAF**), Σ_d(**REBAF**), Σ_r(**REBAF**), Σ_s(**REBAF**). The common idea is that a structure gathers the acceptable elements w.r.t. it. However, in REBAF, acceptability is encoded with the predicates $eAcc$ and $eVal$ whereas in RAF, acceptability is encoded with the predicates Acc and Val.

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 $(S_{\mathcal{I}'} \cup \Gamma_{\mathcal{I}'} \cup \Delta_{\mathcal{I}'}) \subset (S_\mathcal{I} \cup \Gamma_\mathcal{I} \cup \Delta_\mathcal{I})$.

We have the following characterizations:

Proposition 3 *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 conflict-free iff there exists $\mathcal{I}$ model of $\Sigma(\mathbf{REBAF})$ with $S_\mathcal{I} = S$, $\Gamma_\mathcal{I} = \Gamma$ and $\Delta_\mathcal{I} = \Delta$.*
2. *U is admissible iff there exists $\mathcal{I}$ model of $\Sigma_d(\mathbf{REBAF})$ with $S = S_\mathcal{I}$, $\Gamma_\mathcal{I} = \Gamma$ and $\Delta_\mathcal{I} = \Delta$.*
3. *U is complete iff there exists $\mathcal{I}$ model of $\Sigma_d(\mathbf{REBAF}) \cup \Sigma_r(\mathbf{REBAF})$ with $S = S_\mathcal{I}$, $\Gamma_\mathcal{I} = \Gamma$ and $\Delta_\mathcal{I} = \Delta$.*
4. *U is a stable structure iff there exists $\mathcal{I}$ model of $\Sigma_s(\mathbf{REBAF})$ with $S_\mathcal{I} = S$, $\Gamma_\mathcal{I} = \Gamma$ and $\Delta_\mathcal{I} = \Delta$.*
5. *U is a preferred structure iff there exists $\mathcal{I}$ $\subseteq$-maximal model of $\Sigma_d(\mathbf{REBAF})$ with $S_\mathcal{I} = S$, $\Gamma_\mathcal{I} = \Gamma$ and $\Delta_\mathcal{I} = \Delta$.*
6. *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 model of $\Sigma_d(\mathbf{REBAF}) \cup \Sigma_r(\mathbf{REBAF})$.*[19]

The following examples illustrate the above proposition. The first two exemplify the use of formula (17). The third one exemplifies the case of an element which is attacked by a supported and unattacked attack (formulae (12) and (18)). The last two exemplify the case of an element which is attacked by an unattacked but unsupportable attack (formulae (11) and (18)).

Example 6 (cont'd): From $\Sigma_d(\mathbf{REBAF})$ it can be deduced that $eAcc(b) \rightarrow eAcc(a)$ and $eAcc(b) \rightarrow eVal(\alpha)$. That proves that each model of $\Sigma_d(\mathbf{REBAF})$ satisfying $eAcc(b)$ also satisfies $eAcc(a)$ and $eVal(\alpha)$. In other words, given $\mathcal{I}$ a model of

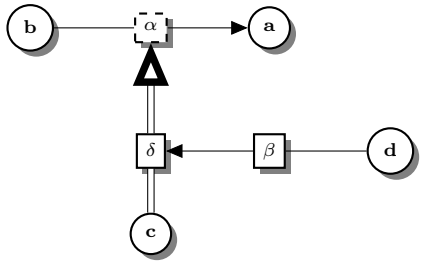
[19] It also holds that U is the grounded structure iff $U = (S_\mathcal{I}, \Gamma_\mathcal{I}, \Delta_\mathcal{I})$ where $\mathcal{I}$ is a $\subseteq$-minimal model of $\Sigma_r(\mathbf{REBAF})$.

$\Sigma_d(\textbf{REBAF})$, if $b \in S_{\mathcal{I}}$ then $a \in S_{\mathcal{I}}$ and $\alpha \in \Delta_{\mathcal{I}}$. That corresponds to the fact that the structure $(\{b\}, \varnothing, \{\alpha\})$ is not admissible.
Moreover, there is a model of $\Sigma_d(\textbf{REBAF})$ satisfying $eAcc(b)$ (and so $eAcc(a)$ and $eVal(\alpha)$). That corresponds to the fact that the structure $(\{a, b\}, \varnothing, \{\alpha\})$ is admissible. □

Example 7 (cont'd): From $\Sigma_d(\textbf{REBAF})$ it can be deduced that $\neg eVal(\alpha)$, $\neg Supp(b)$ and $\neg eAcc(b)$. Moreover there is a model of $\Sigma_d(\textbf{REBAF})$ satisfying $eAcc(a)$. That corresponds to the fact that the unique non-empty admissible structure is $(\{a\}, \varnothing, \varnothing)$.
Note that given $\mathcal{I}$ a model of $\Sigma_d(\textbf{REBAF})$, it holds that $\mathcal{I}$ satisfies $\neg Supp(\alpha)$ and $\neg Supp(b)$. That corresponds to the fact that no admissible structure contains b (resp. α) because b (resp. α) lacks support. □

Example 2 (cont'd): From $\Sigma_d(\textbf{REBAF})$ it can be deduced that $\neg Val(\alpha)$ then $\neg eVal(\alpha)$, $\neg Supp(b)$ and $\neg eAcc(b)$. That corresponds to the fact that no admissible structure contains b (resp. α, though being supported).
Moreover there is a model of $\Sigma_d(\textbf{REBAF})$ satisfying $eAcc(a)$, $eAcc(c)$ and $eVal(\beta)$. That corresponds to the fact that $(\{a, c\}, \varnothing, \{\beta\})$ is an admissible structure. □

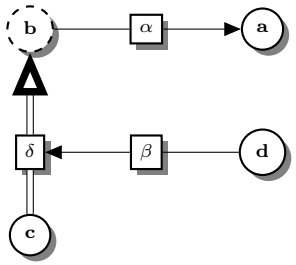
Example 11 *Consider the following argumentation framework.*



From formula (11), it holds that the formula $Acc(a) \rightarrow (UnSupp(\alpha) \vee UnSupp(b))$ *belongs to* $\Sigma_d(\textbf{REBAF})$. *So it can be deduced that* $Acc(a) \rightarrow UnSupp(\alpha)$ *as* b *is prima-facie. Then we can obtain the formula* $eAcc(a) \rightarrow UnSupp(\alpha)$. *Moreover, applying formula (18) yields* $UnSupp(\alpha) \leftrightarrow (eVal(\beta) \wedge eAcc(d))$ *as* δ *and* c *are prima-facie. As a consequence, we obtain* $eAcc(a) \rightarrow (eVal(\beta) \wedge eAcc(d))$.
Applying formula (12) yields $\neg Val(\delta)$, *as* β *and* d *are prima-facie, and as a consequence* $\neg eVal(\delta)$.
Finally, applying formula (17), we obtain $eAcc(a) \rightarrow \neg Supp(\alpha)$ *as* α *is not prima-facie, and as a consequence* $eAcc(a) \rightarrow \neg eVal(\alpha)$.
That corresponds to the fact that if an admissible structure contains a, *then it also contains* d *and* β *and it does not contain* α. *Moreover no admissible structure contains* δ.

From $\Sigma_r(\textbf{REBAF})$ *it can be deduced that* $eAcc(d)$, $eVal(\beta)$ *and* $UnSupp(\alpha) \rightarrow Acc(a)$. *So, from* $\Sigma_d(\textbf{REBAF}) \cup \Sigma_r(\textbf{REBAF})$ *it can be deduced that* $Acc(a)$ *and also* $eAcc(a)$ *as* a *is prima-facie.* $\Sigma_r(\textbf{REBAF})$ *also allows to deduce* $eAcc(b)$ *and* $eAcc(c)$.
That corresponds to the fact that the unique complete structure is $(\{a, b, c, d\}, \{\beta\}, \varnothing)$.
□

Example 12 *Consider the following argumentation framework.*



The same reasoning as the one presented for Ex. 11 can be used, exchanging the role of b *and* α*.*

So from formulae (11) and (18), we obtain the formula $eAcc(a) \rightarrow (eVal(\beta) \wedge eAcc(d))$*.*
Then applying formula (12) trivially yields the formula $\neg eVal(\delta)$*.*
Finally, applying formula (17), we obtain $eAcc(a) \rightarrow \neg eAcc(b)$*.*
That corresponds to the fact that if an admissible structure contains a*, then it also contains* d *and* β *and it does not contain* b*. Moreover no admissible structure contains* δ*.*

Considering $\Sigma_r(\textbf{REBAF}) \cup \Sigma_d(\textbf{REBAF})$*, we obtain* $(\{a, c, d\}, \{\alpha, \beta\}, \varnothing)$ *as the unique complete structure.* □

As said before, a RAF can be considered as a special case of REBAF with no support and all elements being prima-facie. Hence, the logical encoding proposed in Section 6 allows to retrieve the results obtained with Proposition 1.

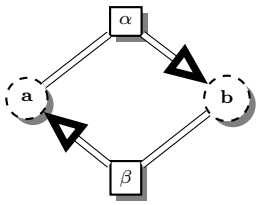
Example 5 (cont'd): As each element is prima-facie, the following formulae can be deduced from $\Sigma(\textbf{REBAF})$:
$\forall x \in (Attack \cup ESupport \cup Arg)\ (Supp(x))$
$\forall x \in Arg\ (Acc(x) \leftrightarrow eAcc(x))$
$\forall x \in (Attack \cup ESupport)\ (Val(x) \leftrightarrow eVal(x))$.
Similarly, from $\Sigma_d(\textbf{REBAF})$, we obtain:
$\forall x \in (Attack \cup ESupport \cup Arg)\ (\neg UnSupp(x))$.
Using these formulae, it is easy to see that $\Sigma(\textbf{REBAF})$ is logically equivalent to the base $\Sigma(\textbf{RAF})$ obtained in Section 3 and that the base $\Sigma_d(\textbf{REBAF})$ (resp. $\Sigma_r(\textbf{REBAF})$) is logically equivalent to the base $\Sigma_d(\textbf{RAF})$ (resp. $\Sigma_r(\textbf{RAF})$) obtained in Section 4. As a consequence it holds that every model of $\Sigma_d(\textbf{REBAF}) \cup \Sigma_r(\textbf{REBAF})$) satisfies $eAcc(a)$, $eAcc(c)$, $eAcc(d)$, $eVal(\alpha)$, $eVal(\delta)$ and falsifies $eAcc(b)$ and $eVal(\beta)$. That corresponds to the unique complete REBAF structure $(\{a, c, d\}, \{\alpha, \delta\}, \varnothing)$, which itself corresponds to the unique RAF structure $(\{a, c, d\}, \{\alpha, \delta\})$. □

6.3 The case of REBAF with support cycles

The logical representation proposed in this paper applies to a restricted variant of REBAF in which interactions are assumed to be binary and there is no cycle of support (see sections 5, 6.1 and 6.2). This restriction allows a direct encoding of the notions

$Sup(U)$ and $UnSupp(U)$. Nevertheless it is interesting to see what happens in the case of a REBAF with support cycles. Let us first consider examples of such a REBAF.

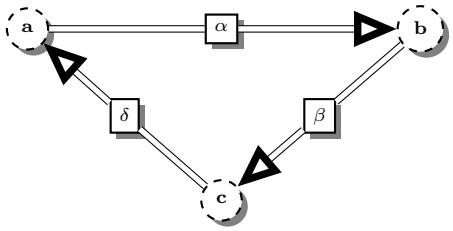
Example 13 *This first example corresponds to an even-length support cycle in which interactions are prima-facie and arguments are not.*



From Σ_{ss} (and so from Σ_d), the following formulae can be entailed: $Supp(a) \leftrightarrow eVal(\beta) \wedge eAcc(b)$ and $Supp(b) \leftrightarrow eVal(\alpha) \wedge eAcc(a)$. There exists a model $\mathcal{I}$ of Σ_d with $S_\mathcal{I} = \{a, b\}$, $\Gamma_\mathcal{I} = \varnothing$ and $\Delta_\mathcal{I} = \{\alpha, \beta\}$. The structure $(S_\mathcal{I}, \Gamma_\mathcal{I}, \Delta_\mathcal{I})$ is not admissible since it is not self-supporting in the sense of Definition 12: there is no chain of support leading to a (resp. to b) rooted in a prima-facie argument. So, Proposition 3 cannot be applied.

□

Example 14 *The second example correspond to an odd-length support cycle in which interactions are prima-facie and arguments are not.*



In this example the same problem appears: there exists a model $\mathcal{I}$ of Σ_d with $S_\mathcal{I} = \{a, b, c\}$, $\Gamma_\mathcal{I} = \varnothing$ and $\Delta_\mathcal{I} = \{\alpha, \beta, \delta\}$ that does not correspond to an admissible structure, since there is no chain of support leading to a (resp. to b, to c) rooted in a prima-facie argument. So Proposition 3 cannot be applied.

□

Example 15 *The third example corresponds to a support loop in which the interaction is prima-facie and the argument is not. And the same problem appears: there is a model of Σ_d corresponding to the structure $(\{a\}, \varnothing, \{\alpha\})$, which is not admissible.*

□

As shown by the above examples, in presence of cycles of support, some of the structures obtained through the models of $\Sigma_d(\mathbf{REBAF})$ are not admissible as there are not self-supporting w.r.t. the definition of $Sup(U)$ given in Definition 12.

A proposal for solving this problem is to exclude those structures using an appropriate selection of models of $\Sigma_d(\mathbf{REBAF})$.

Let U be a structure such that U is self-supporting w.r.t. the definition of $Sup(U)$ given in Definition15. If U is not self-supporting w.r.t. the definition of $Sup(U)$ given in Definition 12, intuitively, there must exist some element (argument or support) x in U such that x cannot be supported without itself. For such an argument (resp. support) x, it should hold that the formula $Supp(x) \rightarrow eAcc(x)$ (resp. $Supp(x) \rightarrow eVal(x)$) is logically entailed by $\Sigma_{ss}(\mathbf{REBAF})$.

Example 13 (cont'd): Consider a model $\mathcal{I}$ of Σ_{ss} with $S_\mathcal{I} = \{a, b\}$, $\Gamma_\mathcal{I} = \varnothing$ and $\Delta_\mathcal{I} = \{\alpha, \beta\}$. It holds that the structure $U_\mathcal{I} = (S_\mathcal{I}, \Gamma_\mathcal{I}, \Delta_\mathcal{I})$ is not self-supporting w.r.t. Definition 12 (though self-supporting w.r.t. Definition15). As shown above, Σ_{ss} entails the formulae $Supp(a) \rightarrow eAcc(b)$, $eAcc(b) \rightarrow Supp(b)$ and $Supp(b) \rightarrow eAcc(a)$. So $U_\mathcal{I}$ contains an argument, a, for which it holds that the formula $Supp(a) \rightarrow eAcc(a)$ is entailed by Σ_{ss}. □

Moreover, if an element (argument or support) x cannot be supported without itself, it cannot be supportable w.r.t. a structure. The above remarks suggest to define "support-founded" interpretations as follows:

Definition 16 *$\mathcal{I}$ is a* support-founded interpretation *iff for each argument (resp. support) x s.t. $\Sigma_{ss}(\mathbf{REBAF})$ entails $Supp(x) \rightarrow eAcc(x)$ (resp. $Supp(x) \rightarrow eVal(x)$), it holds that $\mathcal{I}(eAcc(x)) =$ false (resp. $\mathcal{I}(eVal(x)) =$ false) and $\mathcal{I}(UnSupp(x)) =$ true.*

Then a support-founded model *of $\Sigma_d(\mathbf{REBAF})$ is a support-founded interpretation which is a model of $\Sigma_d(\mathbf{REBAF})$.*

The exact characterization of admissible structures of a given **REBAF** with support cycles by a subclass of models of $\Sigma_d(\mathbf{REBAF})$ is given by the following result.

Proposition 4 *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 $\Sigma_d(\mathbf{REBAF})$ 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 $(\Sigma_d(\mathbf{REBAF}) \cup \Sigma_r(\mathbf{REBAF}))$ 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 $\Sigma_d(\mathbf{REBAF})$ 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 $(\Sigma_d(\mathbf{REBAF}) \cup \Sigma_r(\mathbf{REBAF}))$.*

Let us illustrate the above results on the previous examples:

Example 13 (cont'd): The support-founded models of Σ_d yield all the admissible structures:

$(\varnothing, \varnothing, \varnothing)$
$(\varnothing, \varnothing, \{\alpha\})$
$(\varnothing, \varnothing, \{\beta\})$
$(\varnothing, \varnothing, \{\alpha, \beta\})$

The unique complete structure is $(\varnothing, \varnothing, \{\alpha, \beta\})$.

□

Example 14 (cont'd): The support-founded models of Σ_d yield all the admissible structures:

$(\varnothing, \varnothing, \varnothing)$
$(\varnothing, \varnothing, \{\alpha\})$
$(\varnothing, \varnothing, \{\beta\})$
$(\varnothing, \varnothing, \{\delta\})$
$(\varnothing, \varnothing, \{\alpha, \beta\})$
$(\varnothing, \varnothing, \{\beta, \delta\})$
$(\varnothing, \varnothing, \{\alpha, \delta\})$
$(\varnothing, \varnothing, \{\alpha, \beta, \delta\})$

The unique complete structure is $(\varnothing, \varnothing, \{\alpha, \beta, \delta\})$.

□

Example 15 (cont'd): The support-founded models of Σ_d yield all the admissible structures:

$(\varnothing, \varnothing, \varnothing)$
$(\varnothing, \varnothing, \{\alpha\})$

The unique complete structure is $(\varnothing, \varnothing, \{\alpha\})$.

□

7 Related Works

From the seminal work of Dung [30], several works have proposed to connect abstract argumentation with logic programming (see [25] for recent work and more references). The issue is to find an appropriate encoding of an AF into a logic program P, so that applying logic programming semantics to P enables to capture argumentation semantics of the original AF. Dung [30] has proposed an encoding allowing the capture of (only) grounded and stable semantics. In [25], the encoding allows for the characterization of the standard argumentation semantics (grounded, stable, preferred and complete semantics) through 3-valued models of a logic program.

In the particular case of an AF, the logical representation of [26] is very close to our proposal. This approach uses the logic called CN, which is classical propositional logic augmented with strong negation. An attack (a, b) is encoded in CN by the formula $a \rightarrow Nb$, Nb meaning "b being attacked". So the atom Nb has the same reading as the literal $NAcc(b)$ in our proposal. In [26], an AF is encoded by a CN theory whose models correspond to the AF complete extensions. Indeed, the theory encodes

the requirements defining complete labellings. In contrast, in our proposal, a set of formulae is associated with each standard principle (defence, reinstatement, stability), thus providing a modular encoding of AF semantics.
Note that, again in [26], the CN formulae have been extended in order to encode an AF with joint attacks. Hence, higher-order attacks can be expressed in CN through the reduction of higher-order attacks to first-order joint attacks, as defined in [21]. Following this approach, semantics for RAF could be defined as models of the corresponding CN theory.
Our proposal follows a different path. Starting with RAF acceptability semantics that have been defined in a direct way, in terms of structures, we revisit and encode standard principles with logical formulae, then we characterize semantics in terms of logical models of appropriate theories.

The issue of logical encoding of abstract argumentation has recently been addressed for different other purposes.

In [27,34] acceptance conditions and standard semantics are encoded by first-order logical formulae (given a semantics σ and a set S of arguments, a formula is provided which is satisfiable iff S is a σ-extension). However, the argumentation framework itself is not represented. A similar issue is addressed in [35] with a modal logic, considering that the accessibility relation is the inverse of the attack relation; the same kind of work is presented in [36] using signed theories and QBF formulae; [37] presents algorithms using particular logical notions (minimal correction sets, backbone) in order to compute some semantics (semi-stable and eager); [38] translates complete labellings into logical formulae in order to compute preferred extensions with SAT solvers; [39] proposes a metalevel analysis of the computation problems related to given semantics in order to automatically generate solvers adapted to these problems.

In the more general abstract dialectical framework [40], each argument is associated with a propositional formula which represents the acceptance conditions of the argument. This logical translation enables to capture easily the stable semantics. However, recursive interactions are not taken into account.

[28] proposes a complete framework for handling the dynamics on an AF. A first-order logical language is presented, enabling to describe the structure of an AF, to express incomplete knowledge on an AF and to encode change operations on an AF.

Moreover in the context of the First International Competition on Computational Models of Argumentation (ICCMA), different solvers have been proposed and tested (*e.g.* [41], or [42]). However, in all these works, the attack relation itself is not logically encoded, and neither higher-order attacks, nor supports are taken into account.

8 Conclusion and perspectives

In this work, first we have proposed a logical encoding of argumentation frameworks with higher-order attacks (RAF). Our proposal enables to separate the logical expression of the meaning of an attack (simple or higher-order) and the logical expression of acceptability semantics. These semantics (introduced in [22]) specify the conditions under which the arguments (resp. the attacks) are considered as accepted, directly on the extended framework, without translating the original framework into an AF.

Then, we have been able to characterize the output of a given argumentation framework (under the form of structures following a given semantics) in logical terms (namely as particular models of a logical theory associated with the semantics).

Another feature of our work is its conservative generalization of AF, when d-structures are considered.

In the second part of this paper, using similar ideas and principles as the ones used in the first part, we have considered argumentation frameworks with higher-order attacks and evidential supports (REBAF). We have proposed a logical encoding of REBAF, separating the meaning of attacks and evidential supports (simple or higher-order) and the logical expression of the acceptability semantics introduced in [24]. In the particular case when there is no cycle of support, we have characterized the output of a given REBAF (under the form of structures following a given semantics) in logical terms (namely as models of a logical theory associated with the semantics). Then we have proposed a new kind of models, the support-founded models, for providing characterizations of admissible (resp. complete) structures in the presence of support cycles.

Among future works, we plan to complete the logical characterization of REBAF acceptability semantics. For instance, in the presence of support cycles, a characterization of stable semantics should be provided. Moreover, in a REBAF as defined in Definition 10, the source of an interaction may be a set of arguments and not only one argument. As the current paper only considers a simplified version of REBAF with "binary interactions", it remains to extend this work to REBAF with "joint interactions".

Another line for further research will be to consider other kinds of support (necessary or deductive supports, for instance).

Ongoing work concerns computational issues through the use of logical tools. As a work in that direction, a software has been yet developed (see [43]) that enables to represent a RAF, to express the associated logical theories $\Sigma(\mathbf{RAF})$, $\Sigma_d(\mathbf{RAF})$, ..., and to compute the structures under different semantics. This software should be updated in order to integrate the representation and the treatment for REBAF.

References

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

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

[3] N. Karacapilidis and D. Papadias, "Computer supported argumentation and collaborative decision making: the HERMES system," *Information systems*, vol. 26, no. 4, pp. 259–277, 2001.

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

[5] C. Cayrol and M.-C. Lagasquie-Schiex, "Gradual valuation for bipolar argumentation frameworks," in *Proc. of ECSQARU*, 2005, pp. 366–377.

[6] G. Boella, D. M. Gabbay, L. van der Torre, and S. Villata, "Support in abstract argumentation," in *Proc. of COMMA*, 2010, pp. 111–122.

[7] F. Nouioua and V. Risch, "Bipolar argumentation frameworks with specialized supports," in *Proc. of ICTAI*. IEEE Computer Society, 2010, pp. 215–218.

[8] ——, "Argumentation frameworks with necessities," in *Proc. of SUM*, 2011, pp. 163–176.

[9] N. Oren and T. J. Norman, "Semantics for evidence-based argumentation," in *Proc. of COMMA*, 2008, pp. 276–284.

[10] N. Oren, C. Reed, and M. Luck, "Moving between argumentation frameworks," in *Proc. of COMMA*, 2010, pp. 379–390.

[11] C. Cayrol and M.-C. Lagasquie-Schiex, "Bipolarity in argumentation graphs: towards a better understanding," *Intl. J. of Approximate Reasoning*, vol. 54, no. 7, pp. 876–899, 2013.

[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*, vol. 29, pp. 513–550, 2014.

[13] H. Barringer, D. Gabbay, and J. Woods, "Temporal dynamics of support and attack networks : From argumentation to zoology," in *Mechanizing Mathematical Reasoning. LNAI 2605*. Springer Verlag, 2005, pp. 59–98.

[14] S. Modgil, "Reasoning about preferences in argumentation frameworks," *Artificial Intelligence*, vol. 173, pp. 901–934, 2009.

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

[16] D. M. Gabbay, "Fibring argumentation frames," *Studia Logica*, vol. 93, pp. 231–295, 2009.

[17] S. Villata, G. Boella, D. M. Gabbay, and L. van der Torre, "Modelling defeasible and prioritized support in bipolar argumentation," *AMAI*, vol. 66, no. 1-4, pp. 163–197, 2012.

[18] A. Cohen, S. Gottifredi, A. J. García, and G. R. Simari, "An approach to abstract argumentation with recursive attack and support," *J. Applied Logic*, vol. 13, no. 4, pp. 509–533, 2015.

[19] R. Arisaka and K. Satoh, "Voluntary manslaughter? a case study with meta-argumentation with supports," in *Proc. of JSAI-isAI 2016. LNCS, vol 10247*, 2017, pp. 241–252.

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

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

[22] 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*, vol. 2052. CEUR Workshop Proceedings, 2017.

[23] 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*, 2016, pp. 231–242.

[24] 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*, vol. LNCS 10833. Springer-Verlag, 2018, pp. 150–169.

[25] M. Caminada, S. Sá, J. Alcântara, and W. Dvořák, "On the equivalence between logic programming semantics and argumentation semantics," *Int. Journal of Approximate Reasoning*, vol. 58, pp. 87 – 111, 2015.

[26] D. M. Gabbay and M. Gabbay, "The attack as strong negation, part I," *Logic Journal of the IGPL*, vol. 23, no. 6, pp. 881–941, 2015.

[27] P. Besnard, S. Doutre, and A. Herzig, "Encoding argument graphs in logic," in *Proc of IPMU*, 2014, pp. 345–354.

[28] F. Dupin de Saint-Cyr, P. Bisquert, C. Cayrol, and M.-C. Lagasquie-Schiex, "Argumentation update in YALLA (Yet Another Logic Language for Argumentation)," *Intl. J. of Approximate Reasoning*, vol. 75, pp. 57 – 92, 2016.

[29] C. Cayrol and M.-C. Lagasquie-Schiex, "Logical encoding of argumentation frameworks with higher-order attacks," in *Proc. of ICTAI*. IEEE, 2018.

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

[31] C. Cayrol, J. Fandinno, L. Fariñas del Cerro, and M.-C. Lagasquie-Schiex, "Valid attacks in Argumentation Frameworks with Recursive Attacks (long version)," IRIT, Rapport de recherche IRIT/RR–2019–02–FR, 2019.

[32] R. Demolombe, L. Fariñas del Cerro, and N. Obeid, "Molecular Interaction Automated Maps," in *Proc. of LNMR*, 2013, pp. 31–42.

[33] J. M. Alliot, R. Demolombe, L. Fariñas del Cerro, M. Diéguez, and N. Obeid, "Reasoning on molecular interaction maps," in *Proc. of ESCIM*, 2015, pp. 263–269.

[34] P. Besnard, S. Doutre, V. H. Ho, and D. Longin, "SESAME - A System for Specifying Semantics in Abstract Argumentation," in *Proc. of SAFA*, 2016, pp. 40–51.

[35] D. Grossi, "On the logic of argumentation theory," in *Proc. of AAMAS*, 2010, pp. 409–416.

[36] O. Arieli and M. Caminada, "A QBF-based formalization of abstract argumentation semantics," *Journal of Applied Logic*, vol. 11, no. 2, pp. 229 – 252, 2013.

[37] J. P. Wallner, G. Weissenbacher, and S. Woltran, "Advanced SAT techniques for abstract argumentation," in *Proc. of CLIMA*, 2013, pp. 138–154.

[38] F. Cerutti, P. E. Dunne, M. Giacomin, and M. Vallati, "Computing preferred extensions in abstract argumentation: A SAT-based approach," in *Proc. of TAFA, Revised Selected papers*, 2014, pp. 176–193.

[39] B. Bogaerts, T. Janhunen, and S. Tasharrofi, "Declarative solver development: Case studies," in *Proc. of KR*, 2016, pp. 74–83.

[40] G. Brewka and S. Woltran, "Abstract dialectical frameworks," in *Proc. of KR*, 2010, pp. 102–111.

[41] C. Beierle, F. Brons, and N. Potyka, "A software system using a SAT solver for reasoning under complete, stable, preferred, and grounded argumentation semantics," in *Proc. of KI*, 2015, pp. 241–248.

[42] J. M. Lagniez, E. Lonca, and J. G. Mailly, "Coquiaas: A constraint-based quick abstract argumentation solver," in *Proc. of ICTAI*, 2015, pp. 928–935.

[43] C. Cayrol and M.-C. Lagasquie-Schiex, "The Grafix website." [Online]. Available: http://www.irit.fr/grafix

A Proofs

A.1 Proofs for RAF

Proof of Proposition 1.

1. Let us recall that $\Sigma(RAF)$ includes formulae (1), (2), (3).

   $\Rightarrow$ Assume that the structure $U = (S, \Gamma)$ is conflict-free. Let us define an interpretation $\mathcal{I}$ of $\Sigma(RAF)$ as follows :
   - For all $x \in \mathbf{A} \cup \mathbf{R}$, $\mathcal{I}(Arg(x)) =$ *true* iff $x \in \mathbf{A}$ and $\mathcal{I}(Attack(x)) =$ *true* iff $x \in \mathbf{R}$
   - For all $x \in \mathbf{A}$, $\mathcal{I}(Acc(x)) =$ *true* iff $x \in S$ and $\mathcal{I}(NAcc(x)) =$ *true* iff $\mathcal{I}(Acc(x)) =$ *false*.

- For all $x \in \mathbf{R}$, $\mathcal{I}(Val(x)) =$ *true* iff $x \in \Gamma$.
We have $S_\mathcal{I} = S$ and $\Gamma_\mathcal{I} = \Gamma$. It remains to prove that $\mathcal{I}$ is a model of $\Sigma(RAF)$.
Obviously $\mathcal{I}$ satisfies formula (3).
If $\mathcal{I}$ does not satisfy formula (2), there exist $x \in \mathbf{A}$ and $\alpha \in \mathbf{R}$ such that $t_\alpha = x$, $\mathcal{I}(Val(\alpha)) =$ *true*, $\mathcal{I}(Acc(s_\alpha)) =$ *true* and $\mathcal{I}(NAcc(x)) =$ *false*.
In other words, $\alpha \in \Gamma$, $s_\alpha \in S$ and $x \in S$. That is in contradiction with (S, Γ) being conflict-free.
If $\mathcal{I}$ does not satisfy formula (1), there exist $\alpha, \beta \in \mathbf{R}$ such that $t_\alpha = \beta$, $\mathcal{I}(Val(\alpha)) =$ *true*, $\mathcal{I}(Acc(s_\alpha)) =$ *true* and $\mathcal{I}(Val(\beta)) =$ *true*.
In other words, $\alpha, \beta \in \Gamma$ and $s_\alpha \in S$. That is in contradiction with (S, Γ) being conflict-free.
It follows easily that $\mathcal{I}$ is a model of $\Sigma(RAF)$.

$\Leftarrow$ Let $\mathcal{I}$ be a model of $\Sigma(RAF)$. We prove that the structure $(S_\mathcal{I}, \Gamma_\mathcal{I})$ is conflict-free.
If it is not the case, either there exist $a, b \in S_\mathcal{I}$ and $\alpha \in \Gamma_\mathcal{I}$ with $s_\alpha = a$ and $t_\alpha = b$, or there exist $\alpha, \beta \in \Gamma_\mathcal{I}$ with $s_\alpha \in S_\mathcal{I}$ and $t_\alpha = \beta$.
In the first case, formula (2) is falsified. In the second case, formula (1) is falsified. That is in contradiction with $\mathcal{I}$ being a model of $\Sigma(RAF)$.

2. Let us recall that $\Sigma_d(RAF)$ includes formulae (11), (12).

$\Rightarrow$ Assume that the structure $U = (S, \Gamma)$ is admissible. Due to the proof of the first item, $\exists \mathcal{I}$ model of $\Sigma(RAF)$ with $S_\mathcal{I} = S$ and $\Gamma_\mathcal{I} = \Gamma$ and such that:

- $\forall x \in S, \mathcal{I}(Acc(x)) =$ *true* and $\mathcal{I}(NAcc(x)) =$ *false*;
- $\forall x \in \mathbf{A} \setminus S, \mathcal{I}(Acc(x)) =$ *false* and $\mathcal{I}(NAcc(x)) =$ *true*.
- For all $x \in \mathbf{R}$, $\mathcal{I}(Val(x)) =$ *true* iff $x \in \Gamma$

We have to prove that $\mathcal{I}$ satisfies formulae (11), (12). Let us first consider formula (11). Let $\alpha \in \mathbf{R}$ and $x \in \mathbf{A}$ such that $x = t_\alpha$ and $\mathcal{I}(Acc(x)) =$ *true*. That means that $x \in S$. As U is an admissible structure, we know that x is acceptable wrt U. So there exists $\beta \in \Gamma$ with $s_\beta \in S$ and $t_\beta = s_\alpha$ or $t_\beta = \alpha$. So we have $\mathcal{I}(Acc(s_\beta)) =$ *true* and $\mathcal{I}(Val(\beta)) =$ *true*. We have proved that $\mathcal{I}$ satisfies formula (11). Similarly, it is easy to prove that $\mathcal{I}$ satisfies formula (12).

$\Leftarrow$ Let $\mathcal{I}$ be a model of $\Sigma_d(RAF)$. We have to prove that the structure $U_\mathcal{I} = (S_\mathcal{I}, \Gamma_\mathcal{I})$ is admissible.
As $\Sigma_d(RAF)$ contains $\Sigma(RAF)$, from the first item, we know that the structure $(S_\mathcal{I}, \Gamma_\mathcal{I})$ is conflict-free. Assume that $x \in S_\mathcal{I}$ is the target of an attack α. From formula (11) and $\mathcal{I}(Acc(x)) =$ *true*, it follows that there is $\beta \in \mathbf{R}$ with $t_\beta = s_\alpha$ or $t_\beta = \alpha$, and $\mathcal{I}(Acc(s_\beta)) =$ *true* and $\mathcal{I}(Val(\beta)) =$ *true*. That means that $s_\beta \in S_\mathcal{I}$ and $\beta \in \Gamma_\mathcal{I}$. So either $\alpha \in Inh(U_\mathcal{I})$ or $s_\alpha \in Def(U_\mathcal{I})$. We have proved that x is acceptable w.r.t. $U_\mathcal{I}$. Similarly it is easy to prove that each attack of $\Gamma_\mathcal{I}$ is acceptable w.r.t. $U_\mathcal{I}$.

3. Let us recall that $\Sigma_r(RAF)$ includes formulae (13), (14).

$\Rightarrow$ Assume that the structure $U = (S, \Gamma)$ is complete. Due to the proof of the second item, $\exists \mathcal{I}$ model of $\Sigma_d(RAF)$ with $S_\mathcal{I} = S$ and $\Gamma_\mathcal{I} = \Gamma$ and such that:

- $\forall x \in S, \mathcal{I}(Acc(x)) =$ *true* and $\mathcal{I}(NAcc(x)) =$ *false*;
- $\forall x \in \mathbf{A} \setminus S, \mathcal{I}(Acc(x)) =$ *false* and $\mathcal{I}(NAcc(x)) =$ *true*.
- For all $x \in \mathbf{R}$, $\mathcal{I}(Val(x)) =$ *true* iff $x \in \Gamma$

We have to prove that $\mathcal{I}$ satisfies formulae (13), (14). Let us first consider formula (13). Let $c \in \mathbf{A}$ such that $(\forall \alpha \in Attack)(t_\alpha = c \rightarrow (\exists \beta \in Attack)(t_\beta \in \{s_\alpha, \alpha\} \wedge Val(\beta) \wedge Acc(s_\beta)))$ is satisfied by $\mathcal{I}$. We have to prove that $\mathcal{I}(Acc(c)) =$ *true*. If $\mathcal{I}$ satisfies $(\forall \alpha \in Attack)(t_\alpha = c \rightarrow (\exists \beta \in Attack)(t_\beta \in \{s_\alpha, \alpha\} \wedge Val(\beta) \wedge Acc(s_\beta)))$, $\forall \alpha \in \mathbf{R}$ s.t. $t_\alpha = c$, $\exists \beta \in \mathbf{R}$ s.t. $\mathcal{I}(Val(\beta)) =$ *true* and $\mathcal{I}(Acc(s_\beta)) =$ *true* and $t_\beta = s_\alpha$ or $t_\beta = \alpha$. In other words, due to the definition of $\mathcal{I}$, $\forall \alpha \in \mathbf{R}$ s.t. $t_\alpha = c$, $\exists \beta \in \Gamma$ s.t. $s_\beta \in S$ and $t_\beta = s_\alpha$ or $t_\beta = \alpha$. That exactly means that $\forall \alpha \in \mathbf{R}$ s.t. $t_\alpha = c$, $s_\alpha \in Def(U)$ or $\alpha \in Inh(U)$, or in other words that $c \in Acc(U)$. As U is complete, it follows that $c \in S$, so we obtain $\mathcal{I}(Acc(c)) =$ *true*. Similarly, it is easy to prove that $\mathcal{I}$ satisfies formula (14).

$\Leftarrow$ Let $\mathcal{I}$ be a model of $\Sigma_d(RAF) \cup \Sigma_r(RAF)$. We know that the structure $(S_\mathcal{I}, \Gamma_\mathcal{I})$ is admissible.
It remains to prove that each $x \in \mathbf{A}$ (resp. $x \in \mathbf{R}$) which is acceptable w.r.t. $(S_\mathcal{I}, \Gamma_\mathcal{I})$ belongs to $S_\mathcal{I}$ (resp. $\Gamma_\mathcal{I}$). In other words, we have to prove that each $x \in \mathbf{A}$ (resp. $x \in \mathbf{R}$) which is acceptable w.r.t. $(S_\mathcal{I}, \Gamma_\mathcal{I})$ satisfies $\mathcal{I}(Acc(x)) =$ *true* (resp. $\mathcal{I}(Val(x)) =$ *true*). That follows easily from the fact that $\mathcal{I}$ satisfies formula (13) (resp. (14)) instantiated with $c = x$ (resp. $\delta = x$).

4. Let us recall that $\Sigma_s(RAF)$ includes formulae (15), (16).

$\Rightarrow$ Assume that the structure (S, Γ) is stable. Due to the proof of the first item, $\exists \mathcal{I}$ model of $\Sigma(RAF)$ with $S_\mathcal{I} = S$ and $\Gamma_\mathcal{I} = \Gamma$ and such that:

- $\forall x \in S, \mathcal{I}(Acc(x)) =$ *true* and $\mathcal{I}(NAcc(x)) =$ *false*;
- $\forall x \in \mathbf{A} \setminus S, \mathcal{I}(Acc(x)) =$ *false* and $\mathcal{I}(NAcc(x)) =$ *true*.
- For all $x \in \mathbf{R}$, $\mathcal{I}(Val(x)) =$ *true* iff $x \in \Gamma$

We have to prove that $\mathcal{I}$ satisfies formulae (15), (16). Let us first consider formula (15). Let $c \in \mathbf{A}$ s.t. $\mathcal{I}(\neg Acc(c)) =$ *true*. It means that $c \in \mathbf{A} \setminus S$. As U is stable, we have that $c \in Def(U)$. So there exists $\beta \in \Gamma$ s.t. $t_\beta = c$ and $s_\beta \in S$. Then we have that $\mathcal{I}(Val(\beta)) =$ *true* and $\mathcal{I}(Acc(s_\beta)) =$ *true* which proves that $\mathcal{I}$ satisfies formula (15). Similarly, it is easy to prove that $\mathcal{I}$ satisfies formula (16).

$\Leftarrow$ Let $\mathcal{I}$ be a model of $\Sigma_s(RAF)$. As $\Sigma_s(RAF)$ contains $\Sigma(RAF)$, from the first item, we know that the structure $(S_\mathcal{I}, \Gamma_\mathcal{I})$ is conflict-free. Then the fact that $\mathcal{I}$ satisfies formulae (15), (16) enables to prove that the structure $(S_\mathcal{I}, \Gamma_\mathcal{I})$ satisfies the two following conditions: $\forall a \in \mathbf{A} \setminus S_\mathcal{I}$, a is defeated

w.r.t. $(S_{\mathcal{I}}, \Gamma_{\mathcal{I}})$, and $\forall \alpha \in \mathbf{R} \setminus \Gamma_{\mathcal{I}}$, α is inhibited w.r.t. $(S_{\mathcal{I}}, \Gamma_{\mathcal{I}})$. So the structure $(S_{\mathcal{I}}, \Gamma_{\mathcal{I}})$ is stable.

5. Let $\mathcal{I}$ be an interpretation of a set of formulae Σ. Let $U_{\mathcal{I}}$ denote the structure $(S_{\mathcal{I}}, \Gamma_{\mathcal{I}})$. It is easy to see that $\mathcal{I}$ is a $\subseteq$-maximal model of Σ iff the structure $U_{\mathcal{I}}$ is $\subseteq$-maximal among all the structures of the form $U_{\mathcal{J}} = (S_{\mathcal{J}}, \Gamma_{\mathcal{J}})$, where $\mathcal{J}$ denotes a model of Σ. Then taking $\Sigma = \Sigma_d(RAF)$, it follows that the preferred structures correspond to the structures $U_{\mathcal{I}}$ where $\mathcal{I}$ is a $\subseteq$-maximal model of $\Sigma_d(RAF)$.

6. Let $\mathcal{I}$ be an interpretation of a set of formulae Σ. Let $U_{\mathcal{I}}$ denote the structure $(S_{\mathcal{I}}, \Gamma_{\mathcal{I}})$. It is easy to see that $\mathcal{I}$ is a $\subseteq$-minimal model of Σ iff the structure $U_{\mathcal{I}}$ is $\subseteq$-minimal among all the structures of the form $U_{\mathcal{J}}$, where $\mathcal{J}$ denotes a model of Σ. In particular, it holds for $\Sigma = \Sigma_r(RAF)$.
   From the third item of the current proof, it holds that $\mathcal{I}$ satisfies formulae (13), (14) iff $Acc(U_{\mathcal{I}}) \subseteq (S_{\mathcal{I}} \cup \Gamma_{\mathcal{I}})$. By definition, the grounded structure is the $\subseteq$-minimal conflict-free structure $U = (S, \Gamma)$ satisfying $Acc(U) \subseteq S \cup \Gamma$. It follows that the grounded structure corresponds to the structure $U_{\mathcal{I}}$ where $\mathcal{I}$ is a $\subseteq$-minimal model of $\Sigma_r(RAF)$.

Proof of Proposition 2.

Let $U = (S, \Gamma)$ be a conflict-free structure. Let us consider the model $\mathcal{I}$ of $\Sigma(RAF)$ defined in the first item of the proof of Proposition 1. We have $S = S_{\mathcal{I}}$ and $\Gamma = \Gamma_{\mathcal{I}}$. Let us prove that $\mathcal{I}$ satisfies formula (14) iff $Acc(U_{\mathcal{I}}) \cap \mathbf{R} \subseteq \Gamma_{\mathcal{I}}$. Then, from this result and Proposition 1, it will follow easily that U is a conflict-free (resp. admissible) d-structure iff $\exists \mathcal{I}$ model of $\Sigma(RAF) \cup \{(14)\}$ such that $S_{\mathcal{I}} = S$ and $\Gamma_{\mathcal{I}} = \Gamma$.

Formula (14) writes:

$$(\forall \delta \in Attack)(((\forall \alpha \in Attack)(t_\alpha = \delta \rightarrow (\exists \beta \in Attack)(t_\beta \in \{s_\alpha, \alpha\} \wedge Val(\beta) \wedge Acc(s_\beta)))) \rightarrow Val(\delta))$$

1. Assume that $\mathcal{I}$ satisfies formula (14). Let $\delta \in Acc(U_{\mathcal{I}}) \cap \mathbf{R}$. If δ is not attacked, the formula $(\forall \alpha \in Attack)(t_\alpha = \delta \rightarrow (\exists \beta \in Attack)(t_\beta \in \{s_\alpha, \alpha\} \wedge Val(\beta) \wedge Acc(s_\beta)))$ is trivially true. So, $Val(\delta)$ must also be satisfied by $\mathcal{I}$. That means $\delta \in \Gamma_{\mathcal{I}}$. If δ is attacked, as $\delta \in Acc(U_{\mathcal{I}})$, for each α attacking δ, either $\alpha \in Inh(U_{\mathcal{I}})$ or $s_\alpha \in Def(U_{\mathcal{I}})$. So, for each α attacking δ, there is $\beta \in \Gamma_{\mathcal{I}}$ such that $s_\beta \in S_{\mathcal{I}}$ and $t_\beta \in \{\alpha, s_\alpha\}$. In other words, for each α attacking δ, there is $\beta \in \mathbf{R}$ such that $\mathcal{I}$ satisfies the formulae $Val(\beta)$, $Acc(s_\beta)$ and $t_\beta \in \{\alpha, s_\alpha\}$. So, $Val(\delta)$ must also be satisfied by $\mathcal{I}$. That means $\delta \in \Gamma_{\mathcal{I}}$.

2. Assume that $Acc(U_{\mathcal{I}}) \cap \mathbf{R} \subseteq \Gamma_{\mathcal{I}}$. Let us prove that $\mathcal{I}$ satisfies formula (14). Let δ be an attack such that $\mathcal{I}$ satisfies the formula $(\forall \alpha \in Attack)(t_\alpha = \delta \rightarrow (\exists \beta \in Attack)(t_\beta \in \{s_\alpha, \alpha\} \wedge Val(\beta) \wedge Acc(s_\beta)))$. Then for each attack α attacking δ, there is an attack β such that $\mathcal{I}$ satisfies the formula $t_\beta \in \{s_\alpha, \alpha\} \wedge Val(\beta) \wedge Acc(s_\beta)$. As $\mathcal{I}$ satisfies $Val(\beta) \wedge Acc(s_\beta)$ means that $\beta \in \Gamma_{\mathcal{I}}$ and $s_\beta \in S_{\mathcal{I}}$, we

have that $\delta \in Acc(U_{\mathcal{I}}) \cap \mathbf{R}$. By hypothesis $Acc(U_{\mathcal{I}}) \cap \mathbf{R} \subseteq \Gamma_{\mathcal{I}}$, so $\delta \in \Gamma_{\mathcal{I}}$ and then $\mathcal{I}$ satisfies $Val(\delta)$. We have proved that $\mathcal{I}$ satisfies formula (14).

A.2 Preliminary results for REBAF

For the proofs for REBAF, a new notation and two lemmas are needed.

Notation 1 *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 *[24] Any conflit-free self-supporting structure U satisfies:* $Acc(U) \subseteq \overline{UnAcc(U)} \subseteq \overline{Def(U)}$.

Lemma 2 *Any stable structure U satisfies:* $\overline{Sup(U)} = UnSupp(U)$.

Proof of Lemma2
Assume that U is a stable structure. As U is conflict-free, we already know that $UnSupp(U) \subseteq \overline{Sup(U)}$. It remains to prove the reverse inclusion.
Let us recall that $UnSupp(U) = \overline{Sup(U')}$ where $U' = (\overline{Def_{\mathbf{A}}(U)}, \mathbf{R}_a, \overline{Def_{\mathbf{R}_e}(U)})$.
So we have to prove that $Sup(U') \subseteq Sup(U)$.
Let $x \in Sup(U')$. If $x \in \mathbf{P}$, then $x \in Sup(U)$. If $x \notin \mathbf{P}$, by definition of $Sup(U')$, x is the target of a support α such that $\alpha \in \overline{Def(U)}$, $s_\alpha \in \overline{Def(U)}$, $\alpha \in Sup(U' \setminus \{x\})$ and $s_\alpha \in Sup(U' \setminus \{x\})$. As $U' \setminus \{x\} \subseteq U'$, it follows that $Sup(U' \setminus \{x\}) \subseteq Sup(U')$ (see [24]). So, α (resp. s_α) $\in Sup(U')$. Hence α (resp. s_α) $\notin Unsupp(U)$ by definition of $UnSupp(U)$, and α (resp. s_α) $\notin Def(U)$.
As U is stable, it follows that α (resp. s_α) $\in U$. Any stable structure is self-supporting, so α (resp. s_α) $\in Sup(U)$. Hence we have proved that $x \in Sup(U)$.

A.3 Proofs for REBAF without support cycle

In the case of REBAF without support cycle, Definition 15 is used for $Sup(U)$.

Proof of Proposition 3.

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

1. $\Rightarrow$ Assume that the structure $U = (S, \Gamma, \Delta)$ is conflict-free w.r.t. **REBAF**. Let us define an interpretation $\mathcal{I}$ of Σ(**REBAF**) as follows:

- 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*[20] and
  $\mathcal{I}(UnSupp(x)) =$ *false*.
- For all $x \in \mathbf{A}$, $\mathcal{I}(Acc(x)) =$ *true* iff $x \in S$.
- 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. $\in \Delta$).
- 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*).

It is easy to see that $S_\mathcal{I} = S$, $\Gamma_\mathcal{I} = \Gamma$, $\Delta_\mathcal{I} = \Delta$ and that $\mathcal{I}$ satisfies formulae (3), (1bis), (2bis), (3bis). It remains to prove that $\mathcal{I}$ satisfies the formulae (1), (2).

If $\mathcal{I}$ does not satisfy formula (2), there exist $x \in \mathbf{A}$ and $y \in \mathbf{R}_a$ such that $t_y = x$, $\mathcal{I}(eVal(y)) =$ *true*, $\mathcal{I}(eAcc(s_y)) =$ *true* and $\mathcal{I}(NAcc(x)) =$ *false*. In other words, $y \in \Gamma$, $s_y \in S$ and $x \in S$ (as $\mathcal{I}(NAcc(x)) =$ *false* iff $\mathcal{I}(Acc(x)) =$ *true* iff $x \in S$). That is in contradiction with (S, Γ, Δ) being conflict-free w.r.t. **REBAF**.

If $\mathcal{I}$ does not satisfy formula (1), there exist $x \in \mathbf{R}_a \cup \mathbf{R}_e$ and $y \in \mathbf{R}_a$ such that $t_y = x$, $\mathcal{I}(eVal(y)) =$ *true*, $\mathcal{I}(eAcc(s_y)) =$ *true* and $\mathcal{I}(Val(x)) =$ *true*.

In other words, $y \in \Gamma$, $s_y \in S$ and $x \in \Gamma \cup \Delta$ (as $\mathcal{I}(Val(x)) =$ *true* iff $x \in \Gamma \cup \Delta$). That is in contradiction with (S, Γ, Δ) being conflict-free w.r.t. **REBAF**.

$\Leftarrow$ Let $\mathcal{I}$ be a model of Σ(**REBAF**). We prove that the structure $(S_\mathcal{I}, \Gamma_\mathcal{I}, \Delta_\mathcal{I})$ 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 $s_y \in S_\mathcal{I}$ 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,

[20] Note that a structure $U = (S, \Gamma, \Delta)$ is conflict-free w.r.t. **REBAF** iff it is conflict-free w.r.t. the REBAF $\langle \mathbf{A}, \mathbf{R}_a, \mathbf{R}_e, \mathbf{s}, \mathbf{t}, \mathbf{A} \cup \mathbf{R}_a \cup \mathbf{R}_e \rangle$ as the notion of support has no involvement in conflict-freeness. That means that we may consider a model in which each element of the REBAF receives evidential support.

formula (1) is falsified.
In both cases, there is a contradiction with $\mathcal{I}$ being a model of $\Sigma(\mathbf{REBAF})$.

2. $\Rightarrow$ Assume that the structure $U = (S, \Gamma, \Delta)$ is admissible. Let us define an interpretation $\mathcal{I}$ of $\Sigma_d(\mathbf{REBAF})$. The idea is to define $\mathcal{I}$ by successively adding constraints that $\mathcal{I}$ should satisfy:
   - The assignments for Arg, $Attack$, $ESupport$, $PrimaFacie$, $eAcc$, $eVal$ and $NAcc$ in the interpretation $\mathcal{I}$ are identical to those used in item 1 of the current proof.
   - 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 $x \in \mathbf{R}_a$ (resp. $\in \mathbf{R}_e$), $\mathcal{I}(Val(x)) =$ *true* if and only $x \in \Gamma$ (resp. Δ) or ($x \notin \Gamma$ (resp. Δ), $x \notin Sup(U)$ 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 model of $\Sigma_d(\mathbf{REBAF})$.
   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 $\mathcal{I}(Supp(x)) =$ *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.
   For proving that $\mathcal{I}$ is a model of $\Sigma_d(\mathbf{REBAF})$ it is sufficient to prove that $\mathcal{I}$ satisfies the formulae (1), (2), (3), (1bis), (2bis), (3bis) and (17), (18), (11), (12).
   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 $s_y \in S$ and $y \in \Gamma$. Let us assume that $\mathcal{I}(NAcc(x)) =$ *false*. Then $\mathcal{I}(Acc(x)) =$ *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 $y \in UnAct(U)$, that is y or s_y belongs to $UnAcc(U)$. However, y and s_y being elements of the admissible structure U, due to Lemma 1, we obtain a contradiction. Hence, we have proved that $\mathcal{I}(NAcc(x)) =$ *true* and formula (2) is satisfied by $\mathcal{I}$. Proving that formula (1) is satisfied by $\mathcal{I}$ is similar.
   Let us consider formula (1bis). Let $x \in \mathbf{A} \cup \mathbf{R}_a \cup \mathbf{R}_e$. If $\mathcal{I}(PrimaFacie(x))$ = *true*, $x \in \mathbf{P}$ so $x \in Sup(U)$ and $\mathcal{I}(Supp(x)) =$ *true*. Consider $y \in \mathbf{R}_e$ with $x = t_y$, $\mathcal{I}(eVal(y)) =$ *true* and $\mathcal{I}(eAcc(s_y)) =$ *true*. Then $s_y \in S$ and $y \in \Gamma$. As U is admissible, U is self-supporting, so y and s_y belong to $Sup(U)$. From the definition of $Sup(U)$, it follows that $x \in Sup(U)$ and then $\mathcal{I}(Supp(x)) =$ *true*. Hence we have proved that formula (1bis) is satisfied by $\mathcal{I}$.
   Then we consider formulae (17), (18), (11), (12).
   The fact that $\mathcal{I}$ satisfies formula (17) follows directly from the definition of

$Sup(U)$, the fact that for an argument (resp. a support) x, $\mathcal{I}(eAcc(x)) =$ *true* (resp. $\mathcal{I}(eVal(x)) =$ *true*) iff $x \in S$ (resp. $x \in \Delta$) and the fact that U is admissible hence self-supporting.
The fact that $\mathcal{I}$ satisfies formula (18) follows directly from the definition of $UnSupp(U)$, the fact that $\mathcal{I}(UnSupp(x)) =$ *true* iff $x \in UnSupp(U)$, and the fact that for an argument (resp. an attack) x, $\mathcal{I}(eAcc(x)) =$ *true* (resp. $\mathcal{I}(eVal(x)) =$ *true*) iff $x \in S$ (resp. $x \in \Gamma$).
Let us now consider formula (11). Let $\alpha \in \mathbf{R}_a$ and $x \in \mathbf{A}$ such that $x = t_\alpha$ and $\mathcal{I}(Acc(x)) =$ *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, $\mathcal{I}(eAcc(x)) =$ *true* (resp. $\mathcal{I}(eVal(x)) =$ *true*) iff $x \in S$ (resp. $x \in \Gamma$).
Proving that formula (12) is satisfied by $\mathcal{I}$ is similar.

$\Leftarrow$ Let $\mathcal{I}$ be a model of Σ_d(**REBAF**). We have to prove that the structure $U = (S_\mathcal{I}, \Gamma_\mathcal{I}, \Delta_\mathcal{I})$ is admissible.
As Σ_d(**REBAF**) contains Σ(**REBAF**), from the first item of the current proof, we know that the structure $(S_\mathcal{I}, \Gamma_\mathcal{I}, \Delta_\mathcal{I})$ is conflict-free.
It remains to prove that U is self-supporting (i) and for each element x of the structure, if x is the target of an attack α, then α is unactivable w.r.t. U (ii).
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 y_1 of source x_1 such that $y_1 \in \Delta_\mathcal{I}$ and $x_1 \in \mathbf{S}_\mathcal{I}$. In the first case, it holds that $x \in Sup(U)$. In the other case, it holds that $\mathcal{I}(eAcc(x_1)) =$ *true* and $\mathcal{I}(eVal(y_1)) =$ *true*, and formula (17) can still be used. As it is assumed that there is no cycle of support, this process will end with $y_k \in \mathbf{P}$ and $x_l \in \mathbf{P}$. So $x \in Sup(U)$ by definition of $Sup(U)$. We have proved that U is self-supporting.
Similarly, it can be proved from formula (18) that the following result holds:[21]

$$\mathcal{I}(Unsupp(x)) = \textit{false} \text{ iff } x \notin UnSupp(U) \ (*)$$

Indeed, $\mathcal{I}(Unsupp(x)) =$ *false* iff either $x \in \mathbf{P}$ or there exists $y \in \mathbf{R}_e$ such that $t_y = x$, $\mathcal{I}(UnSupp(y)) =$ *false*, $\mathcal{I}(UnSupp(s_y)) =$ *false*, and neither y nor s_y belongs to $Def(U)$.

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*. From the definition of $\mathcal{I}(eAcc)$, it follows that $\mathcal{I}(Acc(x)) =$ *true*. As $\mathcal{I}$ satisfies formula (11), it follows that either $\mathcal{I}$ satisfies $UnSupp(\alpha)$, or $\mathcal{I}$ satisfies

[21] In order to reuse this result further in the proof, it will be denoted with the expression $(*)$.

$UnSupp(s_\alpha)$, or there exists an attack β targeting α (or s_α) with $\beta \in \Gamma_\mathcal{I}$ and $s_\beta \in S_\mathcal{I}$.
In the first two cases, as $\mathcal{I}$ satisfies formula (18) and following $(*)$, it follows that α (resp. s_α) belongs to $UnSupp(U)$.
In the latter case, it holds that α (resp. s_α) belongs to $Def(U)$. So, we have proved that α is unactivable $wrt\ U$. The same reasoning can be done for $x \in \Gamma_\mathcal{I} \cup \Delta_\mathcal{I}$ using formula (12), hence proving (ii).

3. $\Rightarrow$ Assume that the structure $U = (S, \Gamma, \Delta)$ is complete. Let us define an interpretation $\mathcal{I}$ of $\Sigma_d(\textbf{REBAF}) \cup \Sigma_r(\textbf{REBAF})$ as follows:
   - We keep the same interpretation as the one used in item 2 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 model of $\Sigma_d(\textbf{REBAF}) \cup \Sigma_r(\textbf{REBAF})$.

   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 2 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 $\Sigma_d(\textbf{REBAF})$.
   It remains to 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 $\mathcal{I}(UnSupp(\alpha)) =$ ***true***, or $\mathcal{I}(UnSupp(s_\alpha)) =$ ***true***, or α (or s_α) is attacked by β with $\beta \in \Gamma_\mathcal{I}$ and $s_\beta \in S_\mathcal{I}$. Due to the definition of $\mathcal{I}(UnSupp)$, for each attack α targeting x, either $\alpha \in UnSupp(U)$, or $s_\alpha \in UnSupp(U)$, or α (or s_α) belongs to $Def(U)$. In other words, for each attack α targeting x, $\alpha \in UnAct(U)$, or equivalently $x \in Defended(U)$.
   Now, by definition of $\mathcal{I}(Acc)$, it holds that $\mathcal{I}(Acc(x)) =$ ***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 $\Sigma_d(\textbf{REBAF}) \cup \Sigma_r(\textbf{REBAF})$.

   $\Leftarrow$ Let $\mathcal{I}$ be a 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.
   From item 2 of the current proof, we already know that U is admissible. It remains to prove that $Acc(U)$ is included in $S_\mathcal{I} \cup \Gamma_\mathcal{I} \cup \Delta_\mathcal{I}$.
   As $\mathcal{I}$ satisfies formula (18), we also know that $\mathcal{I}(UnSupp(x)) =$ ***true*** iff $x \in UnSupp(U)$, for all $x \in \mathbf{A} \cup \mathbf{R}_a \cup \mathbf{R}_e$ ($(*)$, again from item 2 of the current proof).

Consider $x \in \mathbf{A} \cap Acc(U)$. It holds that $x \in Sup(U)$ 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 $s_\alpha \in UnSupp(U)$, or α (or s_α) belongs to $Def(U)$ (i.e. α –or s_α– is attacked by $\beta \in U$ with $s_\beta \in U$). So, following $(*)$ and the fact that if an element β (resp. s_β) belongs to 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.

4. $\Rightarrow$ 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 2 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. $\in \mathbf{R}_e$), $\mathcal{I}(Val(x)) = \textit{true}$ if and only $x \in \Gamma$ (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 model of $\Sigma_s(\textbf{REBAF})$.
   Let $x \in S_\mathcal{I}$. By definition, $\mathcal{I}(Acc(x)) = \textit{true}$ and $\mathcal{I}(Supp(x)) = \textit{true}$. By definition of $\mathcal{I}(Acc)$ and $\mathcal{I}(Supp)$, it follows that $x \in Sup(U)$ and ($x \in S$ or $x \notin Def(U)$). As U is conflict-free, it follows that $x \notin UnSupp(U)$ 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 $\mathcal{I}(Acc(x)) = \textit{true}$. As U is stable, U is self-supporting, so $x \in Sup(U)$, then it holds that $\mathcal{I}(Supp(x)) = \textit{true}$. 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.
   For proving that $\mathcal{I}$ is a 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).
   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 $\mathcal{I}(eAcc(s_y) = \textit{true}$. Then $s_y \in S$ 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 $\mathcal{I}(Acc(x)) = \textit{false}$, by definition of $\mathcal{I}(Acc)$ and then $\mathcal{I}(NAcc(x)) = \textit{true}$, 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 the second item of the proof of

Proposition 3 for the admissible case.
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 $s_y \in S$. Hence, there is $y \in \Gamma_{\mathcal{I}}$ with $x = t_y$ and $s_y \in S_{\mathcal{I}}$, 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 $\Sigma_s(\textbf{REBAF})$.

$\Leftarrow$ Let $\mathcal{I}$ be a 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 14, 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 the first item of the current proof, 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 the second item of the proof of Proposition 3 for the admissible case, it can be proved that U is self-supporting and also that $\mathcal{I}(UnSupp(x)) =$ *false* iff $x \notin UnSupp(U)$ $(*)$.
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 $(*)$). In both cases, we have that $x \in UnAcc(U)$. We have proved that U is stable.

5. Let $\mathcal{I}$ be an interpretation of a set of formulae Σ. 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 model of Σ 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 model of Σ. Then taking $\Sigma = \Sigma_d(\textbf{REBAF})$, it follows that the preferred structures correspond to the structures $U_{\mathcal{I}}$ where $\mathcal{I}$ is a $\subseteq$-maximal model of $\Sigma_d(\textbf{REBAF})$.

6. Let $\mathcal{I}$ be an interpretation of a set of formulae Σ. 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 model of Σ iff the structure $U_{\mathcal{I}}$ is $\subseteq$-minimal among all the structures of the form $U_{\mathcal{J}}$, where $\mathcal{J}$ denotes a model of Σ. Taking $\Sigma = \Sigma_d(\textbf{REBAF}) \cup \Sigma_r(\textbf{REBAF})$, it follows that the grounded structure correspond to the structure $U_{\mathcal{I}}$ where $\mathcal{I}$ is a $\subseteq$-minimal model of $\Sigma_d(\textbf{REBAF}) \cup \Sigma_r(\textbf{REBAF})$.

It can also be proved that U is the grounded structure iff $U = (S_{\mathcal{I}}, \Gamma_{\mathcal{I}}, \Delta_{\mathcal{I}})$ where $\mathcal{I}$ is a $\subseteq$-minimal model of $\Sigma_r(\textbf{REBAF})$.

- $\Leftarrow$ Assume that $\mathcal{I}$ is a $\subseteq$-minimal model of $\Sigma_r(\textbf{REBAF})$. Let $U = (S_{\mathcal{I}}, \Gamma_{\mathcal{I}}, \Delta_{\mathcal{I}})$. Let us consider $\mathcal{J}$ a model of $\Sigma_r(\textbf{REBAF})$ such that $S_{\mathcal{I}} = S_{\mathcal{J}}$, $\Gamma_{\mathcal{I}} = \Gamma_{\mathcal{J}}$, $\Delta_{\mathcal{I}} = \Delta_{\mathcal{J}}$, and the set of $eAcc(x)$ satisfied by $\mathcal{J}$ is $\subseteq$-minimal. We first prove that $\mathcal{J}$ is a model of $\Sigma_d(\textbf{REBAF})$. If it is not the case, formula (11) or formula (12) is not satisfied by $\mathcal{J}$. Assume that formula (11) is not satisfied by $\mathcal{J}$ (the reasoning would be similar for formula (12)). There exists $x \in \mathbf{A}$ with $\mathcal{J}(Acc(x)) = \textit{true}$ and an attack α targeting x such that α is unactivable w.r.t. the structure $(S_{\mathcal{J}}, \Gamma_{\mathcal{J}}, \Delta_{\mathcal{J}}) = U$. So, it holds that x is not acceptable w.r.t. U and that $x \notin S_{\mathcal{J}} = S_{\mathcal{I}}$. Then, we can build another interpretation $\mathcal{J}'$ of $\Sigma_r(\textbf{REBAF})$ as follows: $\mathcal{J}'$ satisfies the same literals as $\mathcal{J}$ except that $\mathcal{J}'(Acc(x)) = \textit{false}$. It is easy to see that $\mathcal{J}'$ is still a model of $\Sigma_r(\textbf{REBAF})$ (we just have to reconsider formulae (3), (2bis), (13)). Moreover, as $\mathcal{J}$ and $\mathcal{J}'$ only differ by the interpretation of $Acc(x)$ which is satisfied by $\mathcal{J}$ and not by $\mathcal{J}'$, we obtain a contradiction with the definition of $\mathcal{J}$. So the assumption "formula (11) is not satisfied by $\mathcal{J}$" is false. We have proved that $\mathcal{J}$ is a model of $\Sigma_d(\textbf{REBAF})$, hence a model of $\Sigma_r(\textbf{REBAF}) \cup \Sigma_d(\textbf{REBAF})$.
  Now, we prove that $\mathcal{J}$ is a $\subseteq$-minimal model of $\Sigma_r(\textbf{REBAF}) \cup \Sigma_d(\textbf{REBAF})$. If it is not the case, there exists $\mathcal{J}'$ model of $\Sigma_r(\textbf{REBAF}) \cup \Sigma_d(\textbf{REBAF})$ such that $(S_{\mathcal{J}'} \cup \Gamma_{\mathcal{J}'} \cup \Delta_{\mathcal{J}'}) \subset (S_{\mathcal{J}} \cup \Gamma_{\mathcal{J}} \cup \Delta_{\mathcal{J}}) = (S_{\mathcal{I}} \cup \Gamma_{\mathcal{I}} \cup \Delta_{\mathcal{I}})$. We obtain a contradiction with the fact that $\mathcal{I}$ is a $\subseteq$-minimal model of $\Sigma_r(\textbf{REBAF})$.
  Hence we have proved that the structure $(S_{\mathcal{J}}, \Gamma_{\mathcal{J}}, \Delta_{\mathcal{J}}) = U$ is the grounded structure of **REBAF**, due to item 6 of the current proof.
- $\Rightarrow$ Assume that U is the grounded structure of **REBAF**. There is $\mathcal{I}$ $\subseteq$-minimal model of $\Sigma_r(\textbf{REBAF}) \cup \Sigma_d(\textbf{REBAF})$ with $U = (S_{\mathcal{I}}, \Gamma_{\mathcal{I}}, \Delta_{\mathcal{I}})$. Obviously, $\mathcal{I}$ is a model of $\Sigma_r(\textbf{REBAF})$. If $\mathcal{I}$ is not a $\subseteq$-minimal model of $\Sigma_r(\textbf{REBAF})$, there exists $\mathcal{I}'$ model of $\Sigma_r(\textbf{REBAF})$ with $(S_{\mathcal{I}'} \cup \Gamma_{\mathcal{I}'} \cup \Delta_{\mathcal{I}'}) \subset (S_{\mathcal{I}} \cup \Gamma_{\mathcal{I}} \cup \Delta_{\mathcal{I}})$. As done before, we consider a model $\mathcal{J}'$ of $\Sigma_r(\textbf{REBAF})$ such that $S_{\mathcal{I}'} = S_{\mathcal{J}'}$, $\Gamma_{\mathcal{I}'} = \Gamma_{\mathcal{J}'}$, $\Delta_{\mathcal{I}'} = \Delta_{\mathcal{J}'}$, and the set of $eAcc(x)$ satisfied by $\mathcal{J}'$ is $\subseteq$-minimal. From the first part of the proof of the current item, it holds that $\mathcal{J}'$ is a model of $\Sigma_d(\textbf{REBAF})$. So $\mathcal{J}'$ is a model of $\Sigma_r(\textbf{REBAF}) \cup \Sigma_d(\textbf{REBAF})$ with $(S_{\mathcal{J}'} \cup \Gamma_{\mathcal{J}'} \cup \Delta_{\mathcal{J}'}) \subset (S_{\mathcal{I}} \cup \Gamma_{\mathcal{I}} \cup \Delta_{\mathcal{I}})$. That is in contradiction with the definition of $\mathcal{I}$. Hence, $\mathcal{I}$ is a $\subseteq$-minimal model of $\Sigma_r(\textbf{REBAF})$.

A.4 Proofs for REBAF with support cycles

First of all, let us note that U being a structure, according to Definition 12, $x \in Sup(U)$ iff there is at least one "chain" of supported supports (*i.e.* the support and

its source belong to $U \cap Sup(U)$) leading to x and rooted in prima-facie elements (arguments or supports).
Moreover, as $UnSupp(U) = \overline{Sup(U')}$, it holds that $x \in UnSupp(U)$ iff $x \notin \mathbf{P}$ and there is no chain of supported supports leading to x, rooted in prima-facie elements such that each support (and its source) on the chain is not defeated by U.

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 $s_y \in S \cap Sup(U)$. Then, there exists a support z such that $t_z = x$, $z \in \Delta \cap Sup(U \setminus \{x\})$ and $s_z \in S \cap Sup(U \setminus \{x\})$ and so $x \in Sup(U)$.*

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

Proof of Proposition 4.

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

1. $\Rightarrow$ Assume that the structure $U = (S, \Gamma, \Delta)$ is admissible. Let us build an interpretation $\mathcal{I}$ of $\Sigma_d(\mathbf{REBAF})$ exactly in the same way as for the interpretation used in item 2 of the proof of Proposition 3. Note that in the current case, $Sup(U)$ refers to Definition 12.
   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 $\Sigma_d(\mathbf{REBAF})$.
   The proofs for $S_\mathcal{I} = S$, $\Gamma_\mathcal{I} = \Gamma$, $\Delta_\mathcal{I} = \Delta$ and for the fact that $\mathcal{I}$ satisfies formulae (1), (2), (3), (2bis), (3bis), (11), (12) is the same as the proofs written in item 2 of the proof of Proposition 3. So, it remains to prove that $\mathcal{I}$ satisfies formulae (1bis), (17), (18), and that $\mathcal{I}$ is support-founded.
   Let us first consider formula (17). Let x such that $\mathcal{I}(Supp(x)) =$ *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 $\alpha \in \Delta$, $\alpha \in Sup(U \setminus \{x\})$, $s_\alpha \in S$ and $s_\alpha \in Sup(U \setminus \{x\})$. 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 $\mathcal{I}(eAcc(s_\alpha)) =$ *true* and $\mathcal{I}(eVal(\alpha)) =$ *true*. Hence formula (17) is satisfied by $\mathcal{I}$.
   Let us consider formula (1bis). In the case when $\mathcal{I}(PrimaFacie(x)) =$ *true*, $x \in \mathbf{P}$, so $x \in Sup(U)$, hence $\mathcal{I}(Supp(x)) =$ *true*. Then let us consider the case when x is the target of a support y such that $\mathcal{I}(eAcc(s_y)) =$

true and $\mathcal{I}(eVal(y)) =$ *true*. We have to prove that $\mathcal{I}(Supp(x)) =$ *true*. As $S_{\mathcal{I}} = S$ and $\Delta_{\mathcal{I}} = \Delta$ it holds that $y \in \Delta$ and $s_y \in S$. Moreover, as U is admissible, U is self-supporting, so y and s_y belong to $Sup(U)$. From Lemma 3, it follows that $x \in Sup(U)$ hence $\mathcal{I}(Supp(x)) =$ *true*. So formula (1bis) is satisfied by $\mathcal{I}$.
Let us now consider formula (18) and consider x such that $\mathcal{I}(UnSupp(x)) =$ *true*. By definition of $\mathcal{I}(UnSupp)$, $x \in UnSupp(U) = \overline{Sup(U')}$ (where $U' = (\overline{Def_{\mathbf{A}}(U)}, \mathbf{R}_a, \overline{Def_{\mathbf{R}_e}(U)})$). 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 is defeated by U, or the support or its source 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 is defeated by $(S_{\mathcal{I}}, \Gamma_{\mathcal{I}}, \Delta_{\mathcal{I}}) = U$, or the support or its source 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 12, it holds that $x \notin Sup(U')$, that is $x \in UnSupp(U)$ and so $\mathcal{I}(UnSupp(x)) =$ *true*.

Finally, we have to prove that $\mathcal{I}$ is support-founded. Let $x \in \mathbf{A}$ such that $\Sigma_{ss}($**REBAF**$)$ entails $Supp(x) \rightarrow eAcc(x)$. We have to prove that $\mathcal{I}(eAcc(x)) =$ *false* and $\mathcal{I}(UnSupp(x)) =$ *true*, that is by definition of $\mathcal{I}$, $x \notin S_{\mathcal{I}}$ (i) and $x \in UnSupp(U)$ (ii). We prove the contrapositive.

- For (i), let us assume that $x \in S_{\mathcal{I}}$. We must prove that $\Sigma_{ss}($**REBAF**$)$ does not entail $Supp(x) \rightarrow eAcc(x)$ or equivalently that there is a model $\mathcal{J}$ of $\Sigma_{ss}($**REBAF**$)$ s.t. $\mathcal{J}(Supp(x)) =$ *true* and $\mathcal{J}(Acc(x)) =$ *false*. As $S_{\mathcal{I}} = S$, and U is self-supporting, it holds that $x \in Sup(U)$. Hence, either $x \in \mathbf{P}$ or there is a is a chain of supported supports leading to x, rooted in prima-facie elements and which does not contain x. Then let us consider an interpretation $\mathcal{J}$ defined exactly as $\mathcal{I}$ except that $\mathcal{J}(Acc(x)) =$ *false*. If $x \in \mathbf{P}$, obviously, $\mathcal{J}$ is a model of $\Sigma_{ss}($**REBAF**$)$. In the other case, using the set considered above, it holds that $\mathcal{J}$ satisfies formula (17). Indeed, any argument y in the set being different from x, it holds that $y \in S_{\mathcal{I}}$ iff $\mathcal{J}(eAcc(y)) =$ *true*.

- For (ii), let us assume that $x \notin UnSupp(U)$, or equivalently $x \in Sup(U')$. Hence, either $x \in \mathbf{P}$ or there is a is a chain of supported supports leading to x, rooted in prima-facie elements and which does not contain x. The difference with (i) is that here elements of the chain are not supported by U, but only not defeated by U. However, as for (i), it is possible to build an interpretation $\mathcal{J}$ such that $\mathcal{J}(Supp(x)) =$ *true*, $\mathcal{J}(Acc(x)) =$ *false*, and $\mathcal{J}$ is a model of $\Sigma_{ss}($**REBAF**$)$. The idea is to define $\mathcal{J}$ such that for each y different from x, $\mathcal{J}(Acc(y)) =$ *true* if y is not defeated by U.

The proof for $x \in \mathbf{R}_e$ is similar.

$\Leftarrow$ Let $\mathcal{I}$ be a support-founded model of $\Sigma_d(\textbf{REBAF})$. We have to prove that the structure $U = (S_\mathcal{I}, \Gamma_\mathcal{I}, \Delta_\mathcal{I})$ is admissible.
As the notion of support has no involvement in conflict-freeness, the proof used in the first item of Proposition 3 can still be used for proving that U is conflict-free.
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 y_1 of source x_1 such that $y_1 \in \Delta_\mathcal{I}$ and $x_1 \in \mathbf{S}_\mathcal{I}$. In the first case, it holds that $x \in Sup(U)$. In the other case, it holds that $\mathcal{I}(eAcc(x_1)) =$ *true* and $\mathcal{I}(eVal(y_1)) =$ *true*, and formula (17) can still be used, thus enabling to build a chain of supports. As U is finite, this process will end with $y_k \in \mathbf{P}$ and either $x_l \in \mathbf{P}$ or $x_l = x_i$ with $i < l$. And so it can still be proved that $x \in Sup(U)$ w.r.t. Definition 15. Assume that $x \notin Sup(U)$ w.r.t. Definition 12. Then it holds that some x_i is equal to x. $\Sigma_{ss}(\textbf{REBAF})$ (using formulae (17) and (2bis)) enables to prove $Supp(x) \rightarrow Acc(x_1) \wedge Supp(x_1) \wedge Val(y_1) \wedge Supp(y_1)$ and similar formulae for all the other elements involved in the chain of supports. It follows that $\Sigma_{ss}(\textbf{REBAF})$ enables to prove $Supp(x) \rightarrow Acc(x_i) = Acc(x)$. That is in contradiction with the fact that $\mathcal{I}$ is support-founded. So we have proved that $x \in Sup(U)$ (w.r.t. Definition 12), 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 s_α) with $\beta \in \Delta_\mathcal{I}$ and $s_\beta \in S_\mathcal{I}$, or $\mathcal{I}$ satisfies $UnSupp(\alpha)$ (or $\mathcal{I}$ satisfies $UnSupp(s_\alpha)$).

- In the first case, it holds that α (resp. s_α) belongs to $Def(U)$.
- In the second case, we prove that α (resp. 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 chain of supports leading to x, rooted in prima-facie elements such that each support (and its source) in the chain 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 chain 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. $\Rightarrow$ Assume that the structure $U = (S, \Gamma, \Delta)$ is complete. Let us build an

interpretation $\mathcal{I}$ of $\Sigma_d(\textbf{REBAF}) \cup \Sigma_r(\textbf{REBAF})$ exactly in the same way as for the interpretation used in item 3 of the proof of Proposition 3. Moreover, the proofs used in this item can still be used to prove that $S_{\mathcal{I}} = S$, $\Gamma_{\mathcal{I}} = \Gamma$, $\Delta_{\mathcal{I}} = \Delta$ and $\mathcal{I}$ 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 the item 1 of the current proof can be used as U is self-supporting.

$\Leftarrow$ 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 $S_{\mathcal{I}} \cup \Gamma_{\mathcal{I}} \cup \Delta_{\mathcal{I}}$.
The proof written in item 3 of the proof of Proposition 3 can be used if we prove that $x \in \mathbf{A} \cap Sup(U)$ implies $\mathcal{I}(Supp(x)) = \textit{true}$ and that $x \in UnSupp(U)$ implies $\mathcal{I}(UnSupp(x)) = \textit{true}$.
Let us consider $x \in \mathbf{A} \cap Sup(U)$. From the definition of U, the definition of $Sup(U)$ and the fact that $\mathcal{I}$ satisfies formula (1bis), it follows that $\mathcal{I}(Supp(x)) = \textit{true}$.
Then we prove that if $x \in UnSupp(U)$ then $\mathcal{I}(UnSupp(x)) = \textit{true}$, or equivalently if $\mathcal{I}(UnSupp(x)) = \textit{false}$, then $x \in Sup(U')$. Consider x with $\mathcal{I}(UnSupp(x)) = \textit{false}$. From formula (18), it holds that $x \in \mathbf{P}$ or x is the target of a support y_1 of source x_1 such that y_1 and x_1 are not defeated by U (i), and $\mathcal{I}(UnSupp(y_1)) = \textit{false}$ (ii) and $\mathcal{I}(UnSupp(x_1)) = \textit{false}$ (iii).
If $x \in \mathbf{P}$, it holds that $x \in Sup(U')$. Assume that $x \notin \mathbf{P}$. With (ii) (resp. (iii)), and formula (18) again, a chain of supports can be built such that each element z on the chain (support or source of support) satisfies $\mathcal{I}(UnSupp(z)) = \textit{false}$. Assume that the process that builds this chain cannot end with prima-facie elements. That means that for at least one element z_k, it holds that z_k cannot receive support without itself, so $\mathcal{I}(UnSupp(z_k)) = \textit{true}$, as $\mathcal{I}$ is support-founded. There is a contradiction (as each element z on the chain satisfies $\mathcal{I}(UnSupp(z)) = \textit{false}$). So, we have obtained a chain of supports leading to x, rooted in prima-facie elements, such that each element on the chain receives no attack from U. By definition of U' it follows that $x \in Sup(U')$.
The proof is similar for any support or attack in $Acc(U)$.

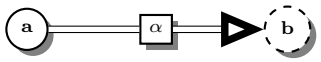
3. The proof is similar as in the case of REBAF without support cycle.

4. The proof is similar as in the case of REBAF without support cycle.

B Examples

B.1 Examples of REBAF without cycles of support

Example 6 (cont'd): Consider a REBAF with only one support which is prima-facie and whose source is also prima-facie.



$\Sigma = \{$
$Supp(a)\ Acc(a) \rightarrow eAcc(a)$
$eAcc(a) \rightarrow Acc(a)$
$eAcc(a) \rightarrow Supp(a)$
$Supp(b)\ Acc(b) \rightarrow eAcc(b)$
$eAcc(b) \rightarrow Acc(b)$
$eAcc(b) \rightarrow Supp(b)$
$Supp(\alpha)\ Val(\alpha) \rightarrow eVal(\alpha)$
$eVal(\alpha) \rightarrow Val(\alpha)$
$eVal(\alpha) \rightarrow Supp(\alpha)$
$\rightarrow Supp(a)$
$\rightarrow Supp(\alpha)$
$eVal(\alpha)\ eAcc(a) \rightarrow Supp(b)\ \}$
$\Sigma_{ss} = \Sigma \cup \{$
$Supp(b) \rightarrow eVal(\alpha)$
$Supp(b) \rightarrow eAcc(a)$
$UnSupp(a) \rightarrow$
$UnSupp(\alpha) \rightarrow$
$UnSupp(b) \rightarrow UnSupp(a)\ UnSupp(\alpha)$
$UnSupp(\alpha) \rightarrow UnSupp(b)$
$UnSupp(a) \rightarrow UnSupp(b)\ \}$
$\Sigma_d = \Sigma_{ss} \cup \varnothing$
$\Sigma_r = \Sigma_{ss} \cup \{$
$\rightarrow Acc(a)$
$\rightarrow Acc(b)$
$\rightarrow Val(\alpha)\ \}$
$\Sigma_s = \Sigma_{ss} \cup \{$
$\rightarrow Acc(a)$
$\rightarrow Acc(b)$
$\rightarrow Val(\alpha)$
$\rightarrow Supp(a)\ UnSupp(a)$
$\rightarrow Supp(b)\ UnSupp(b)$
$\rightarrow Supp(\alpha)\ UnSupp(\alpha)\ \}$

In this case, since α is not attacked, it can be valid and since it is also prima-facie, it is supported and so it is e-valid. As consequence of this, b can be considered as e-accepted (not attacked and supported). So the structure $U = \langle \{a, b\}, \varnothing, \{\alpha\} \rangle$ is an admissible structure among several ones.

Prop 3 produces the following structures:

- Conflict-free structures: all structures are conflict-free.
- Admissible structures:

```
[]
[alpha]
[a]
[a, alpha]
[a, b, alpha]
```

- Preferred structures:

```
[a, b, alpha]
```

- Grounded structure:

```
[a, b, alpha]
```

- Complete structures:

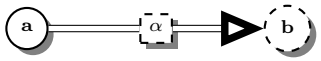
```
[a, b, alpha]
```

- Stable structures:

```
[a, b, alpha]
```

□

Example 7 (cont'd): Consider a REBAF with only one support which is not prima-facie but whose source is prima-facie.



$\Sigma = \{$
$Supp(a)\ Acc(a) \rightarrow eAcc(a)$
$eAcc(a) \rightarrow Acc(a)$
$eAcc(a) \rightarrow Supp(a)$
$Supp(b)\ Acc(b) \rightarrow eAcc(b)$
$eAcc(b) \rightarrow Acc(b)$
$eAcc(b) \rightarrow Supp(b)$
$Supp(\alpha)\ Val(\alpha) \rightarrow eVal(\alpha)$
$eVal(\alpha) \rightarrow Val(\alpha)$
$eVal(\alpha) \rightarrow Supp(\alpha)$
$\rightarrow Supp(a)$
$eVal(\alpha)\ eAcc(a) \rightarrow Supp(b)\ \}$
$\Sigma_{ss} = \Sigma \cup \{$
$Supp(b) \rightarrow eVal(\alpha)$
$Supp(b) \rightarrow eAcc(a)$
$Supp(\alpha) \rightarrow$
$UnSupp(a) \rightarrow$
$\rightarrow UnSupp(\alpha)$

$UnSupp(b) \rightarrow UnSupp(a)\ UnSupp(\alpha)$
$UnSupp(\alpha) \rightarrow UnSupp(b)$
$UnSupp(a) \rightarrow UnSupp(b)\ \}$
$\Sigma_d = \Sigma_{ss} \cup \varnothing$
$\Sigma_r = \Sigma_{ss} \cup \{$
$\rightarrow Acc(a)$
$\rightarrow Acc(b)$
$\rightarrow Val(\alpha)\ \}$
$\Sigma_s = \Sigma_{ss} \cup \{$
$\rightarrow Acc(a)$
$\rightarrow Acc(b)$
$\rightarrow Val(\alpha)$
$\rightarrow Supp(a)\ UnSupp(a)$
$\rightarrow Supp(b)\ UnSupp(b)$
$\rightarrow Supp(\alpha)\ UnSupp(\alpha)\ \}$

In this case, α is not attacked so it can be valid but it is not supported and so it is not e-valid. As consequence of this, even if b can be considered as accepted, it is not e-accepted. So the only admissible structures are $U = \langle \varnothing, \varnothing, \varnothing \rangle$ and $U = \langle \{a\}, \varnothing, \varnothing \rangle$.

Prop 3 produces the following structures:

- Conflict-free structures: all structures are conflict-free.
- Admissible structures:

```
[]
[a]
```

- Preferred structures:

```
[a]
```

- Grounded structure:

```
[a]
```

- Complete structures:

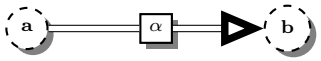
```
[a]
```

- Stable structures:

```
[a]
```

□

Example 8 (cont'd): Consider a REBAF with only one support which is prima-facie but whose source is not prima-facie.



$\Sigma = \{$
$Supp(a)\ Acc(a) \rightarrow eAcc(a)$
$eAcc(a) \rightarrow Acc(a)$
$eAcc(a) \rightarrow Supp(a)$
$Supp(b)\ Acc(b) \rightarrow eAcc(b)$
$eAcc(b) \rightarrow Acc(b)$
$eAcc(b) \rightarrow Supp(b)$
$Supp(\alpha)\ Val(\alpha) \rightarrow eVal(\alpha)$
$eVal(\alpha) \rightarrow Val(\alpha)$
$eVal(\alpha) \rightarrow Supp(\alpha)$
$\rightarrow Supp(\alpha)$
$eVal(\alpha)\ eAcc(a) \rightarrow Supp(b)\ \}$
$\Sigma_{ss} = \Sigma \cup \{$
$Supp(b) \rightarrow eVal(\alpha)$
$Supp(b) \rightarrow eAcc(a)$
$Supp(a) \rightarrow$
$UnSupp(\alpha) \rightarrow$
$\rightarrow UnSupp(a)$
$UnSupp(b) \rightarrow UnSupp(a)\ UnSupp(\alpha)$
$UnSupp(\alpha) \rightarrow UnSupp(b)$
$UnSupp(a) \rightarrow UnSupp(b)\ \}$
$\Sigma_d = \Sigma_{ss} \cup \varnothing$
$\Sigma_r = \Sigma_{ss} \cup \{$
$\rightarrow Acc(a)$
$\rightarrow Acc(b)$
$\rightarrow Val(\alpha)\ \}$
$\Sigma_s = \Sigma_{ss} \cup \{$
$\rightarrow Acc(a)$
$\rightarrow Acc(b)$
$\rightarrow Val(\alpha)$
$\rightarrow Supp(a)\ UnSupp(a)$
$\rightarrow Supp(b)\ UnSupp(b)$
$\rightarrow Supp(\alpha)\ UnSupp(\alpha)\ \}$

In this case, α is e-valid (not attacked and supported) but a is not supported. As consequence of this, b is not e-accepted. Here, the only admissible structures are $U = \langle \varnothing, \varnothing, \varnothing \rangle$ and $U = \langle \varnothing, \varnothing, \{\alpha\} \rangle$.

Prop 3 produces the following structures:

- Conflict-free structures: all structures are conflict-free.
- Admissible structures:

```
[]
[alpha]
```

- Preferred structures:

  `[alpha]`

- Grounded structure:

  `[alpha]`

- Complete structures:

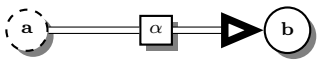
  `[alpha]`

- Stable structures:

  `[alpha]`

□

Example 9 (cont'd): Consider a REBAF with only one support which is prima-facie whose target is also prima-facie but whose source is not prima-facie.



$\Sigma = \{$
$Supp(a)\ Acc(a) \rightarrow eAcc(a)$
$eAcc(a) \rightarrow Acc(a)$
$eAcc(a) \rightarrow Supp(a)$
$Supp(b)\ Acc(b) \rightarrow eAcc(b)$
$eAcc(b) \rightarrow Acc(b)$
$eAcc(b) \rightarrow Supp(b)$
$Supp(\alpha)\ Val(\alpha) \rightarrow eVal(\alpha)$
$eVal(\alpha) \rightarrow Val(\alpha)$
$eVal(\alpha) \rightarrow Supp(\alpha)$
$\rightarrow Supp(\alpha)$
$\rightarrow Supp(b)\ \}$
$\Sigma_{ss} = \Sigma \cup \{$
$Supp(a) \rightarrow$
$UnSupp(\alpha) \rightarrow$
$\rightarrow UnSupp(a)$
$UnSupp(b) \rightarrow\ \}$
$\Sigma_d = \Sigma_{ss} \cup \varnothing$
$\Sigma_r = \Sigma_{ss} \cup \{$
$\rightarrow Acc(a)$
$\rightarrow Acc(b)$
$\rightarrow Val(\alpha)\ \}$
$\Sigma_s = \Sigma_{ss} \cup \{$
$\rightarrow Acc(a)$

$\rightarrow Acc(b)$
$\rightarrow Val(\alpha)$
$\rightarrow Supp(a)\ UnSupp(a)$
$\rightarrow Supp(b)\ UnSupp(b)$
$\rightarrow Supp(\alpha)\ UnSupp(\alpha)$ }

In this case, α and b are e-valid (not attacked and supported) but a is not supported. Here, there are 4 admissible structures: $U = \langle \varnothing, \varnothing, \varnothing \rangle$, $U = \langle \varnothing, \varnothing, \{\alpha\} \rangle$, $U = \langle \{b\}, \varnothing, \varnothing \rangle$ and $U = \langle \{b\}, \varnothing, \{\alpha\} \rangle$.

Prop 3 produces the following structures:

- Conflict-free structures: all structures are conflict-free.
- Admissible structures:

```
[]
[b]
[b, alpha]
[alpha]
```

- Preferred structures:

```
[b, alpha]
```

- Grounded structure:

```
[b, alpha]
```

- Complete structures:

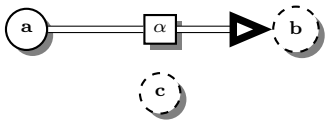
```
[b, alpha]
```

- Stable structures:

```
[b, alpha]
```

□

Example 16 *Consider the REBAF presented in Ex. 6 in which an additional non prima-facie argument is added without any other interaction.*



$\Sigma = \{$
$Supp(a)\ Acc(a) \rightarrow eAcc(a)$
$eAcc(a) \rightarrow Acc(a)$
$eAcc(a) \rightarrow Supp(a)$

$Supp(b)\ Acc(b) \rightarrow eAcc(b)$
$eAcc(b) \rightarrow Acc(b)$
$eAcc(b) \rightarrow Supp(b)$
$Supp(c)\ Acc(c) \rightarrow eAcc(c)$
$eAcc(c) \rightarrow Acc(c)$
$eAcc(c) \rightarrow Supp(c)$
$Supp(\alpha)\ Val(\alpha) \rightarrow eVal(\alpha)$
$eVal(\alpha) \rightarrow Val(\alpha)$
$eVal(\alpha) \rightarrow Supp(\alpha)$
$\rightarrow Supp(a)$
$\rightarrow Supp(\alpha)$
$eVal(\alpha)\ eAcc(a) \rightarrow Supp(b)\ \}$

$\Sigma_{ss} = \Sigma \cup \{$
$Supp(b) \rightarrow eVal(\alpha)$
$Supp(b) \rightarrow eAcc(a)$
$Supp(c) \rightarrow$
$UnSupp(a) \rightarrow$
$UnSupp(\alpha) \rightarrow$
$\rightarrow UnSupp(c)$
$UnSupp(b) \rightarrow UnSupp(a)\ UnSupp(\alpha)$
$UnSupp(\alpha) \rightarrow UnSupp(b)$
$UnSupp(a) \rightarrow UnSupp(b)\ \}$

$\Sigma_d = \Sigma_{ss} \cup \varnothing$

$\Sigma_r = \Sigma_{ss} \cup \{$
$\rightarrow Acc(a)$
$\rightarrow Acc(b)$
$\rightarrow Acc(c)$
$\rightarrow Val(\alpha)\ \}$

$\Sigma_s = \Sigma_{ss} \cup \{$
$\rightarrow Acc(a)$
$\rightarrow Acc(b)$
$\rightarrow Acc(c)$
$\rightarrow Val(\alpha)$
$\rightarrow Supp(a)\ UnSupp(a)$
$\rightarrow Supp(b)\ UnSupp(b)$
$\rightarrow Supp(c)\ UnSupp(c)$
$\rightarrow Supp(\alpha)\ UnSupp(\alpha)\ \}$

In this case, c cannot be e-accepted since it is not supported. An admissible structure is $U = \langle \{a, b\}, \varnothing, \{\alpha\} \rangle$.

Prop 3 produces the following structures:

- *Conflict-free structures: all structures are conflict-free.*
- *Admissible structures:*

[ ]

```
[alpha]
[a, alpha]
[a]
[a, b, alpha]
```

- *Preferred structures:*

```
[a, b, alpha]
```

- *Grounded structure:*

```
[a, b, alpha]
```

- *Complete structures:*

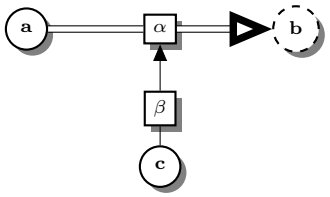
```
[a, b, alpha]
```

- *Stable structures:*

```
[a, b, alpha]
```

□

Example 2 (cont'd): Consider now a more complex REBAF with a support that is the target of an attack.



$\Sigma = \{$
$Supp(a)\ Acc(a) \rightarrow eAcc(a)$
$eAcc(a) \rightarrow Acc(a)$
$eAcc(a) \rightarrow Supp(a)$
$Supp(b)\ Acc(b) \rightarrow eAcc(b)$
$eAcc(b) \rightarrow Acc(b)$
$eAcc(b) \rightarrow Supp(b)$
$Supp(c)\ Acc(c) \rightarrow eAcc(c)$
$eAcc(c) \rightarrow Acc(c)$
$eAcc(c) \rightarrow Supp(c)$
$Supp(\alpha)\ Val(\alpha) \rightarrow eVal(\alpha)$
$eVal(\alpha) \rightarrow Val(\alpha)$
$eVal(\alpha) \rightarrow Supp(\alpha)$
$Supp(\beta)\ Val(\beta) \rightarrow eVal(\beta)$
$eVal(\beta) \rightarrow Val(\beta)$

$eVal(\beta) \rightarrow Supp(\beta)$
$Val(\alpha)\ eVal(\beta)\ eAcc(c) \rightarrow$
$\rightarrow Supp(a)$
$\rightarrow Supp(\alpha)$
$eVal(\alpha)\ eAcc(a) \rightarrow Supp(b)$
$\rightarrow Supp(c)$
$\rightarrow Supp(\beta)\ \}$

$\Sigma_{ss} = \Sigma \cup \{$
$Supp(b) \rightarrow eVal(\alpha)$
$Supp(b) \rightarrow eAcc(a)$
$UnSupp(a) \rightarrow$
$UnSupp(\alpha) \rightarrow$
$UnSupp(c) \rightarrow$
$UnSupp(\beta) \rightarrow$
$UnSupp(b) \rightarrow eVal(\beta)\ UnSupp(a)\ UnSupp(\alpha)$
$UnSupp(b) \rightarrow eAcc(c)\ UnSupp(a)\ UnSupp(\alpha)$
$UnSupp(\alpha) \rightarrow UnSupp(b)$
$UnSupp(a) \rightarrow UnSupp(b)$
$eVal(\beta)\ eAcc(c) \rightarrow UnSupp(b)\ \}$

$\Sigma_d = \Sigma_{ss} \cup \{$
$Val(\alpha) \rightarrow UnSupp(c)\ UnSupp(\beta)\ \}$

$\Sigma_r = \Sigma_{ss} \cup \{$
$\rightarrow Acc(a)$
$\rightarrow Acc(b)$
$\rightarrow Acc(c)$
$UnSupp(\beta) \rightarrow Val(\alpha)$
$UnSupp(c) \rightarrow Val(\alpha)$
$\rightarrow Val(\beta)\ \}$

$\Sigma_s = \Sigma_{ss} \cup \{$
$\rightarrow Acc(a)$
$\rightarrow Acc(b)$
$\rightarrow Acc(c)$
$\rightarrow eVal(\beta)\ Val(\alpha)$
$\rightarrow eAcc(c)\ Val(\alpha)$
$\rightarrow Val(\beta)$
$\rightarrow Supp(a)\ UnSupp(a)$
$\rightarrow Supp(b)\ UnSupp(b)$
$\rightarrow Supp(c)\ UnSupp(c)$
$\rightarrow Supp(\alpha)\ UnSupp(\alpha)$
$\rightarrow Supp(\beta)\ UnSupp(\beta)\ \}$

In the case, α is not valid since it is not defended. So, even if α is supported, it is not e-valid and is not able to support b. So b is not e-accepted. So neither α, nor b can belong to an admissible structure. In this example, there is only one complete, preferred and stable structure: $(\{a, c\}, \{\beta\}, \varnothing)$.

Prop 3 produces the following structures:

- Conflict-free structures: all structures are conflict-free except those containing together c, α and β (so 4 structures are not conflict-free).
- Admissible structures:

```
[]
[c]
[beta]
[c, beta]
[a, c, beta]
[a]
[a, c]
[a, beta]
```

- Preferred structures:

```
[a, c, beta]
```

- Grounded structure:

```
[a, c, beta]
```

- Complete structures:

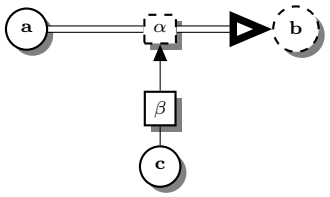
```
[a, c, beta]
```

- Stable structures:

```
[a, c, beta]
```

□

Example 17 *Consider the same example as Ex. 2 but with the support that is not prima-facie.*



$\Sigma = \{$
$Supp(a)\ Acc(a) \rightarrow eAcc(a)$
$eAcc(a) \rightarrow Acc(a)$
$eAcc(a) \rightarrow Supp(a)$
$Supp(b)\ Acc(b) \rightarrow eAcc(b)$
$eAcc(b) \rightarrow Acc(b)$

$eAcc(b) \rightarrow Supp(b)$
$Supp(c)\ Acc(c) \rightarrow eAcc(c)$
$eAcc(c) \rightarrow Acc(c)$
$eAcc(c) \rightarrow Supp(c)$
$Supp(\alpha)\ Val(\alpha) \rightarrow eVal(\alpha)$
$eVal(\alpha) \rightarrow Val(\alpha)$
$eVal(\alpha) \rightarrow Supp(\alpha)$
$Supp(\beta)\ Val(\beta) \rightarrow eVal(\beta)$
$eVal(\beta) \rightarrow Val(\beta)$
$eVal(\beta) \rightarrow Supp(\beta)$
$Val(\alpha)\ eVal(\beta)\ eAcc(c) \rightarrow$
$\rightarrow Supp(a)$
$eVal(\alpha)\ eAcc(a) \rightarrow Supp(b)$
$\rightarrow Supp(c)$
$\rightarrow Supp(\beta)\ \}$

$\Sigma_{ss} = \Sigma \cup \{$
$Supp(b) \rightarrow eVal(\alpha)$
$Supp(b) \rightarrow eAcc(a)$
$Supp(\alpha) \rightarrow$
$UnSupp(a) \rightarrow$
$\rightarrow UnSupp(\alpha)$
$UnSupp(c) \rightarrow$
$UnSupp(\beta) \rightarrow$
$UnSupp(b) \rightarrow eVal(\beta)\ UnSupp(a)\ UnSupp(\alpha)$
$UnSupp(b) \rightarrow eAcc(c)\ UnSupp(a)\ UnSupp(\alpha)$
$UnSupp(\alpha) \rightarrow UnSupp(b)$
$UnSupp(a) \rightarrow UnSupp(b)$
$eVal(\beta)\ eAcc(c) \rightarrow UnSupp(b)\ \}$

$\Sigma_d = \Sigma_{ss} \cup \{$
$Val(\alpha) \rightarrow UnSupp(c)\ UnSupp(\beta)\ \}$

$\Sigma_r = \Sigma_{ss} \cup \{$
$\rightarrow Acc(a)$
$\rightarrow Acc(b)$
$\rightarrow Acc(c)$
$UnSupp(\beta) \rightarrow Val(\alpha)$
$UnSupp(c) \rightarrow Val(\alpha)$
$\rightarrow Val(\beta)\ \}$

$\Sigma_s = \Sigma_{ss} \cup \{$
$\rightarrow Acc(a)$
$\rightarrow Acc(b)$
$\rightarrow Acc(c)$
$\rightarrow eVal(\beta)\ Val(\alpha)$
$\rightarrow eAcc(c)\ Val(\alpha)$
$\rightarrow Val(\beta)$
$\rightarrow Supp(a)\ UnSupp(a)$
$\rightarrow Supp(b)\ UnSupp(b)$

$\rightarrow Supp(c)\ UnSupp(c)$
$\rightarrow Supp(\alpha)\ UnSupp(\alpha)$
$\rightarrow Supp(\beta)\ UnSupp(\beta)\ \}$

In the case, α is not valid since it is not defended. Moreover α is not supported. So it is not e-valid and is not able to support b. So b is not e-accepted (even if it is accepted). So neither α, nor b can belong to an admissible structure. In this example, as in Ex. 2, there is only one complete, preferred and stable structure: $(\{a, c\}, \{\beta\}, \varnothing)$.

Prop 3 produces the following structures:

- *Conflict-free structures: all structures are conflict-free except those containing together c, α and β (so 4 structures are not conflict-free).*
- *Admissible structures:*

```
[]
[c]
[c, beta]
[beta]
[a, c, beta]
[a]
[a, c]
[a, beta]
```

- *Preferred structures:*

```
[a, c, beta]
```

- *Grounded structure:*

```
[a, c, beta]
```

- *Complete structures:*

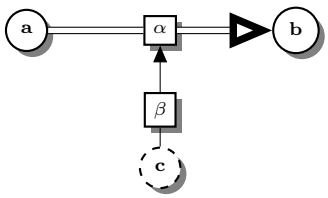
```
[a, c, beta]
```

- *Stable structures:*

```
[a, c, beta]
```

□

Example 18 *Consider another version of Example 2 in which all elements are prima-facie except c.*



$$
\begin{aligned}
\Sigma = \{ & \\
& Supp(a)\ Acc(a) \rightarrow eAcc(a) \\
& eAcc(a) \rightarrow Acc(a) \\
& eAcc(a) \rightarrow Supp(a) \\
& Supp(b)\ Acc(b) \rightarrow eAcc(b) \\
& eAcc(b) \rightarrow Acc(b) \\
& eAcc(b) \rightarrow Supp(b) \\
& Supp(c)\ Acc(c) \rightarrow eAcc(c) \\
& eAcc(c) \rightarrow Acc(c) \\
& eAcc(c) \rightarrow Supp(c) \\
& Supp(\alpha)\ Val(\alpha) \rightarrow eVal(\alpha) \\
& eVal(\alpha) \rightarrow Val(\alpha) \\
& eVal(\alpha) \rightarrow Supp(\alpha) \\
& Supp(\beta)\ Val(\beta) \rightarrow eVal(\beta) \\
& eVal(\beta) \rightarrow Val(\beta) \\
& eVal(\beta) \rightarrow Supp(\beta) \\
& Val(\alpha)\ eVal(\beta)\ eAcc(c) \rightarrow \\
& \rightarrow Supp(a) \\
& \rightarrow Supp(b) \\
& \rightarrow Supp(\alpha) \\
& \rightarrow Supp(\beta)\ \}
\end{aligned}
$$

$$
\begin{aligned}
\Sigma_{ss} = \Sigma \cup \{ & \\
& Supp(c) \rightarrow \\
& UnSupp(a) \rightarrow \\
& UnSupp(b) \rightarrow \\
& UnSupp(\alpha) \rightarrow \\
& UnSupp(\beta) \rightarrow \\
& \rightarrow UnSupp(c)\ \}
\end{aligned}
$$

$$
\begin{aligned}
\Sigma_{d} = \Sigma_{ss} \cup \{ & \\
& Val(\alpha) \rightarrow UnSupp(c)\ UnSupp(\beta)\ \}
\end{aligned}
$$

$$
\begin{aligned}
\Sigma_{r} = \Sigma_{ss} \cup \{ & \\
& \rightarrow Acc(a) \\
& \rightarrow Acc(b) \\
& \rightarrow Acc(c) \\
& UnSupp(\beta) \rightarrow Val(\alpha) \\
& UnSupp(c) \rightarrow Val(\alpha) \\
& \rightarrow Val(\beta)\ \}
\end{aligned}
$$

$$
\begin{aligned}
\Sigma_{s} = \Sigma_{ss} \cup \{ & \\
& \rightarrow Acc(a) \\
& \rightarrow Acc(b) \\
& \rightarrow Acc(c) \\
& \rightarrow eVal(\beta)\ Val(\alpha) \\
& \rightarrow eAcc(c)\ Val(\alpha) \\
& \rightarrow Val(\beta) \\
& \rightarrow Supp(a)\ UnSupp(a) \\
& \rightarrow Supp(b)\ UnSupp(b)
\end{aligned}
$$

$\rightarrow Supp(c)\ UnSupp(c)$
$\rightarrow Supp(\alpha)\ UnSupp(\alpha)$
$\rightarrow Supp(\beta)\ UnSupp(\beta)\ \}$

Prop 3 produces the following structures:

- *Conflict-free structures: all structures are conflict-free except those containing together c, α and β (so 4 structures are not conflict-free).*
- *Admissible structures:*

```
[]
[b]
[b, alpha]
[alpha]
[alpha, beta]
[beta]
[b, beta]
[b, alpha, beta]
[a, alpha, beta]
[a]
[a, b]
[a, b, alpha]
[a, alpha]
[a, beta]
[a, b, beta]
[a, b, alpha, beta]
```

- *Preferred structures:*

```
[a, b, alpha, beta]
```

- *Grounded structure:*

```
[a, b, alpha, beta]
```

- *Complete structures:*

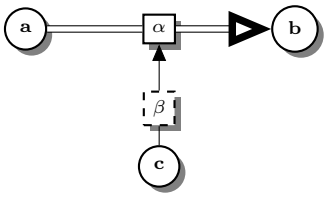
```
[a, b, alpha, beta]
```

- *Stable structures:*

```
[a, b, alpha, beta]
```

□

Example 19 *Consider another version of Example 2 in which all elements are prima-facie except β.*



$\Sigma = \{$
$Supp(a)\ Acc(a) \rightarrow eAcc(a)$
$eAcc(a) \rightarrow Acc(a)$
$eAcc(a) \rightarrow Supp(a)$
$Supp(b)\ Acc(b) \rightarrow eAcc(b)$
$eAcc(b) \rightarrow Acc(b)$
$eAcc(b) \rightarrow Supp(b)$
$Supp(c)\ Acc(c) \rightarrow eAcc(c)$
$eAcc(c) \rightarrow Acc(c)$
$eAcc(c) \rightarrow Supp(c)$
$Supp(\alpha)\ Val(\alpha) \rightarrow eVal(\alpha)$
$eVal(\alpha) \rightarrow Val(\alpha)$
$eVal(\alpha) \rightarrow Supp(\alpha)$
$Supp(\beta)\ Val(\beta) \rightarrow eVal(\beta)$
$eVal(\beta) \rightarrow Val(\beta)$
$eVal(\beta) \rightarrow Supp(\beta)$
$Val(\alpha)\ eVal(\beta)\ eAcc(c) \rightarrow$
$\rightarrow Supp(a)$
$\rightarrow Supp(b)$
$\rightarrow Supp(c)$
$\rightarrow Supp(\alpha)\ \}$
$\Sigma_{ss} = \Sigma \cup \{$
$Supp(\beta) \rightarrow$
$UnSupp(a) \rightarrow$
$UnSupp(b) \rightarrow$
$UnSupp(c) \rightarrow$
$UnSupp(\alpha) \rightarrow$
$\rightarrow UnSupp(\beta)\ \}$
$\Sigma_d = \Sigma_{ss} \cup \{$
$Val(\alpha) \rightarrow UnSupp(c)\ UnSupp(\beta)\ \}$
$\Sigma_r = \Sigma_{ss} \cup \{$
$\rightarrow Acc(a)$
$\rightarrow Acc(b)$
$\rightarrow Acc(c)$
$UnSupp(\beta) \rightarrow Val(\alpha)$
$UnSupp(c) \rightarrow Val(\alpha)$
$\rightarrow Val(\beta)\ \}$

$\Sigma_s = \Sigma_{ss} \cup \{$
$\quad \rightarrow Acc(a)$
$\quad \rightarrow Acc(b)$
$\quad \rightarrow Acc(c)$
$\quad \rightarrow eVal(\beta)\ Val(\alpha)$
$\quad \rightarrow eAcc(c)\ Val(\alpha)$
$\quad \rightarrow Val(\beta)$
$\quad \rightarrow Supp(a)\ UnSupp(a)$
$\quad \rightarrow Supp(b)\ UnSupp(b)$
$\quad \rightarrow Supp(c)\ UnSupp(c)$
$\quad \rightarrow Supp(\alpha)\ UnSupp(\alpha)$
$\quad \rightarrow Supp(\beta)\ UnSupp(\beta)\ \}$

Prop 3 produces the following structures:

- *Conflict-free structures: all structures are conflict-free except those containing together c, α and β (so 4 structures are not conflict-free).*
- *Admissible structures:*

```
[]
[b]
[b, c]
[c]
[b, c, alpha]
[alpha]
[b, alpha]
[c, alpha]
[a, c, alpha]
[a]
[a, b]
[a, b, c]
[a, c]
[a, b, c, alpha]
[a, alpha]
[a, b, alpha]
```

- *Preferred structures:*

```
[a, b, c, alpha]
```

- *Grounded structure:*

```
[a, b, c, alpha]
```

- *Complete structures:*

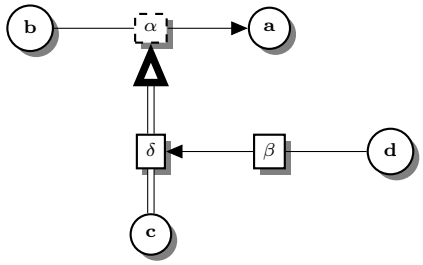
```
[a, b, c, alpha]
```

- *Stable structures:*

```
[a, b, c, alpha]
```

□

Example 11 (cont'd):
Consider the following REBAF.



$\Sigma = \{$
$Supp(a)\ Acc(a) \rightarrow eAcc(a)$
$eAcc(a) \rightarrow Acc(a)$
$eAcc(a) \rightarrow Supp(a)$
$Supp(b)\ Acc(b) \rightarrow eAcc(b)$
$eAcc(b) \rightarrow Acc(b)$
$eAcc(b) \rightarrow Supp(b)$
$Supp(c)\ Acc(c) \rightarrow eAcc(c)$
$eAcc(c) \rightarrow Acc(c)$
$eAcc(c) \rightarrow Supp(c)$
$Supp(d)\ Acc(d) \rightarrow eAcc(d)$
$eAcc(d) \rightarrow Acc(d)$
$eAcc(d) \rightarrow Supp(d)$
$Supp(\alpha)\ Val(\alpha) \rightarrow eVal(\alpha)$
$eVal(\alpha) \rightarrow Val(\alpha)$
$eVal(\alpha) \rightarrow Supp(\alpha)$
$Supp(\beta)\ Val(\beta) \rightarrow eVal(\beta)$
$eVal(\beta) \rightarrow Val(\beta)$
$eVal(\beta) \rightarrow Supp(\beta)$
$Supp(\delta)\ Val(\delta) \rightarrow eVal(\delta)$
$eVal(\delta) \rightarrow Val(\delta)$
$eVal(\delta) \rightarrow Supp(\delta)$
$Val(\delta)\ eVal(\beta)\ eAcc(d) \rightarrow$
$eVal(\alpha)\ eAcc(b) \rightarrow NAcc(a)$
$NAcc(a)\ Acc(a) \rightarrow$
$\rightarrow Supp(a)$
$\rightarrow Supp(b)$
$\rightarrow Supp(c)$
$\rightarrow Supp(d)$

$\to Supp(\beta)$
$\to Supp(\delta)$
$eVal(\delta)\ eAcc(c) \to Supp(\alpha)\ \}$

$\Sigma_{ss} = \Sigma \cup \{$
$Supp(\alpha) \to eVal(\delta)$
$Supp(\alpha) \to eAcc(c)$
$UnSupp(a) \to$
$UnSupp(b) \to$
$UnSupp(c) \to$
$UnSupp(d) \to$
$UnSupp(\beta) \to$
$UnSupp(\delta) \to$
$UnSupp(\alpha) \to eVal(\beta)\ UnSupp(c)\ UnSupp(\delta)$
$UnSupp(\alpha) \to eAcc(d)\ UnSupp(c)\ UnSupp(\delta)$
$UnSupp(\delta) \to UnSupp(\alpha)$
$UnSupp(c) \to UnSupp(\alpha)$
$eVal(\beta)\ eAcc(d) \to UnSupp(\alpha)\ \}$

$\Sigma_d = \Sigma_{ss} \cup \{$
$Acc(a) \to UnSupp(b)\ UnSupp(\alpha)$
$Val(\delta) \to UnSupp(d)\ UnSupp(\beta)\ \}$

$\Sigma_r = \Sigma_{ss} \cup \{$
$UnSupp(\alpha) \to Acc(a)$
$UnSupp(b) \to Acc(a)$
$\to Acc(b)$
$\to Acc(c)$
$\to Acc(d)$
$UnSupp(\beta) \to Val(\delta)$
$UnSupp(d) \to Val(\delta)$
$\to Val(\alpha)$
$\to Val(\beta)\ \}$

$\Sigma_s = \Sigma_{ss} \cup \{$
$\to eVal(\alpha)\ Acc(a)$
$\to eAcc(b)\ Acc(a)$
$\to Acc(b)$
$\to Acc(c)$
$\to Acc(d)$
$\to eVal(\beta)\ Val(\delta)$
$\to eAcc(d)\ Val(\delta)$
$\to Val(\alpha)$
$\to Val(\beta)$
$\to Supp(a)\ UnSupp(a)$
$\to Supp(b)\ UnSupp(b)$
$\to Supp(c)\ UnSupp(c)$
$\to Supp(d)\ UnSupp(d)$
$\to Supp(\alpha)\ UnSupp(\alpha)$
$\to Supp(\beta)\ UnSupp(\beta)$

$\rightarrow Supp(\delta)\ UnSupp(\delta)\ \}$

Prop 3 produces the following structures:

- Conflict-free structures: 98 structures are conflict-free among the 128 possible structures; are not conflict-free structures containing together a, b and α, or those containing d, β, δ.
- Admissible structures:

```
[]
[b]
[b, c]
[c]
[b, c, d]
[d]
[b, d]
[c, d]
[c, beta]
[beta]
[b, beta]
[b, c, beta]
[c, d, beta]
[d, beta]
[b, d, beta]
[b, c, d, beta]
[a, c, d, beta]
[a, d, beta]
[a, b, d, beta]
[a, b, c, d, beta]
```

- Preferred structures:

```
[a, b, c, d, beta]
```

- Grounded structure:

```
[a, b, c, d, beta]
```

- Complete structures:

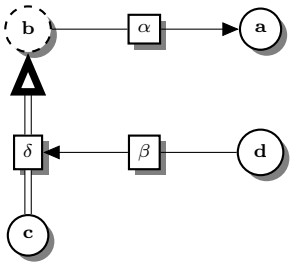
```
[a, b, c, d, beta]
```

- Stable structures:

```
[a, b, c, d, beta]
```

□

Example 12 (cont'd):
Consider the following REBAF.



$\Sigma = \{$
$Supp(a)\ Acc(a) \rightarrow eAcc(a)$
$eAcc(a) \rightarrow Acc(a)$
$eAcc(a) \rightarrow Supp(a)$
$Supp(b)\ Acc(b) \rightarrow eAcc(b)$
$eAcc(b) \rightarrow Acc(b)$
$eAcc(b) \rightarrow Supp(b)$
$Supp(c)\ Acc(c) \rightarrow eAcc(c)$
$eAcc(c) \rightarrow Acc(c)$
$eAcc(c) \rightarrow Supp(c)$
$Supp(d)\ Acc(d) \rightarrow eAcc(d)$
$eAcc(d) \rightarrow Acc(d)$
$eAcc(d) \rightarrow Supp(d)$
$Supp(\alpha)\ Val(\alpha) \rightarrow eVal(\alpha)$
$eVal(\alpha) \rightarrow Val(\alpha)$
$eVal(\alpha) \rightarrow Supp(\alpha)$
$Supp(\beta)\ Val(\beta) \rightarrow eVal(\beta)$
$eVal(\beta) \rightarrow Val(\beta)$
$eVal(\beta) \rightarrow Supp(\beta)$
$Supp(\delta)\ Val(\delta) \rightarrow eVal(\delta)$
$eVal(\delta) \rightarrow Val(\delta)$
$eVal(\delta) \rightarrow Supp(\delta)$
$Val(\delta)\ eVal(\beta)\ eAcc(d) \rightarrow$
$eVal(\alpha)\ eAcc(b) \rightarrow NAcc(a)$
$NAcc(a)\ Acc(a) \rightarrow$
$\rightarrow Supp(a)$
$eVal(\delta)\ eAcc(c) \rightarrow Supp(b)$
$\rightarrow Supp(c)$
$\rightarrow Supp(d)$
$\rightarrow Supp(\beta)$
$\rightarrow Supp(\delta)$
$\rightarrow Supp(\alpha)\ \}$
$\Sigma_{ss} = \Sigma \cup \{$

$Supp(b) \rightarrow eVal(\delta)$
$Supp(b) \rightarrow eAcc(c)$
$UnSupp(a) \rightarrow$
$UnSupp(b) \rightarrow eVal(\beta)\ UnSupp(c)\ UnSupp(\delta)$
$UnSupp(b) \rightarrow eAcc(d)\ UnSupp(c)\ UnSupp(\delta)$
$UnSupp(\delta) \rightarrow UnSupp(b)$
$UnSupp(c) \rightarrow UnSupp(b)$
$eVal(\beta)\ eAcc(d) \rightarrow UnSupp(b)$
$UnSupp(c) \rightarrow$
$UnSupp(d) \rightarrow$
$UnSupp(\alpha) \rightarrow$
$UnSupp(\beta) \rightarrow$
$UnSupp(\delta) \rightarrow \}$

$\Sigma_d = \Sigma_{ss} \cup \{$
$Acc(a) \rightarrow UnSupp(b)\ UnSupp(\alpha)$
$Val(\delta) \rightarrow UnSupp(d)\ UnSupp(\beta)\ \}$

$\Sigma_r = \Sigma_{ss} \cup \{$
$UnSupp(\alpha) \rightarrow Acc(a)$
$UnSupp(b) \rightarrow Acc(a)$
$\rightarrow Acc(b)$
$\rightarrow Acc(c)$
$\rightarrow Acc(d)$
$UnSupp(\beta) \rightarrow Val(\delta)$
$UnSupp(d) \rightarrow Val(\delta)$
$\rightarrow Val(\alpha)$
$\rightarrow Val(\beta)\ \}$

$\Sigma_s = \Sigma_{ss} \cup \{$
$\rightarrow eVal(\alpha)\ Acc(a)$
$\rightarrow eAcc(b)\ Acc(a)$
$\rightarrow Acc(b)$
$\rightarrow Acc(c)$
$\rightarrow Acc(d)$
$\rightarrow eVal(\beta)\ Val(\delta)$
$\rightarrow eAcc(d)\ Val(\delta)$
$\rightarrow Val(\alpha)$
$\rightarrow Val(\beta)$
$\rightarrow Supp(a)\ UnSupp(a)$
$\rightarrow Supp(b)\ UnSupp(b)$
$\rightarrow Supp(c)\ UnSupp(c)$
$\rightarrow Supp(d)\ UnSupp(d)$
$\rightarrow Supp(\alpha)\ UnSupp(\alpha)$
$\rightarrow Supp(\beta)\ UnSupp(\beta)$
$\rightarrow Supp(\delta)\ UnSupp(\delta)\ \}$

Prop 3 produces the following structures:

- Conflict-free structures: there are 98 among the 128 possible structures; are not

conflict-free structures that contain together a, b and α, or those containing d, β, δ.

- Admissible structures:

```
[]
[c]
[c, d]
[d]
[c, d, alpha]
[alpha]
[c, alpha]
[d, alpha]
[c, alpha, beta]
[beta]
[c, beta]
[alpha, beta]
[d, alpha, beta]
[d, beta]
[c, d, beta]
[c, d, alpha, beta]
[a, c, d, alpha, beta]
[a, d, beta]
[a, c, d, beta]
[a, d, alpha, beta]
```

- Preferred structures:

```
[a, c, d, alpha, beta]
```

- Grounded structure:

```
[a, c, d, alpha, beta]
```

- Complete structures:

```
[a, c, d, alpha, beta]
```

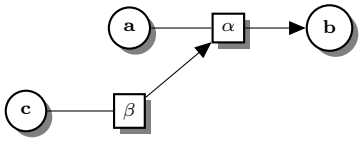
- Stable structures:

```
[a, c, d, alpha, beta]
```

□

B.2 Examples of RAF (with or without cycles)

Example 4 (cont'd):



$\Sigma = \{$
$Supp(a)\ Acc(a) \rightarrow eAcc(a)$
$eAcc(a) \rightarrow Acc(a)$
$eAcc(a) \rightarrow Supp(a)$
$Supp(b)\ Acc(b) \rightarrow eAcc(b)$
$eAcc(b) \rightarrow Acc(b)$
$eAcc(b) \rightarrow Supp(b)$
$Supp(c)\ Acc(c) \rightarrow eAcc(c)$
$eAcc(c) \rightarrow Acc(c)$
$eAcc(c) \rightarrow Supp(c)$
$Supp(\alpha)\ Val(\alpha) \rightarrow eVal(\alpha)$
$eVal(\alpha) \rightarrow Val(\alpha)$
$eVal(\alpha) \rightarrow Supp(\alpha)$
$Supp(\beta)\ Val(\beta) \rightarrow eVal(\beta)$
$eVal(\beta) \rightarrow Val(\beta)$
$eVal(\beta) \rightarrow Supp(\beta)$
$Val(\alpha)\ eVal(\beta)\ eAcc(c) \rightarrow$
$eVal(\alpha)\ eAcc(a) \rightarrow NAcc(b)$
$NAcc(b)\ Acc(b) \rightarrow$
$\rightarrow Supp(a)$
$\rightarrow Supp(b)$
$\rightarrow Supp(c)$
$\rightarrow Supp(\alpha)$
$\rightarrow Supp(\beta)\ \}$
$\Sigma_{ss} = \Sigma \cup \{$
$UnSupp(a) \rightarrow$
$UnSupp(b) \rightarrow$
$UnSupp(c) \rightarrow$
$UnSupp(\alpha) \rightarrow$
$UnSupp(\beta) \rightarrow \}$
$\Sigma_d = \Sigma_{ss} \cup \{$
$Val(\alpha) \rightarrow UnSupp(c)\ UnSupp(\beta)$
$Acc(b) \rightarrow eVal(\beta)\ UnSupp(a)\ UnSupp(\alpha)$
$Acc(b) \rightarrow eAcc(c)\ UnSupp(a)\ UnSupp(\alpha)\ \}$
$\Sigma_r = \Sigma_{ss} \cup \{$
$\rightarrow Acc(a)$

$\rightarrow Acc(c)$
$UnSupp(\alpha) \rightarrow Acc(b)$
$UnSupp(a) \rightarrow Acc(b)$
$eVal(\beta)\ eAcc(c) \rightarrow Acc(b)$
$\rightarrow Val(\beta)$
$UnSupp(\beta) \rightarrow Val(\alpha)$
$UnSupp(c) \rightarrow Val(\alpha)\ \}$

$\Sigma_s = \Sigma_{ss} \cup \{$
$\rightarrow Acc(a)$
$\rightarrow eVal(\alpha)\ Acc(b)$
$\rightarrow eAcc(a)\ Acc(b)$
$\rightarrow Acc(c)$
$\rightarrow eVal(\beta)\ Val(\alpha)$
$\rightarrow eAcc(c)\ Val(\alpha)$
$\rightarrow Val(\beta)$
$\rightarrow Supp(a)\ UnSupp(a)$
$\rightarrow Supp(b)\ UnSupp(b)$
$\rightarrow Supp(c)\ UnSupp(c)$
$\rightarrow Supp(\alpha)\ UnSupp(\alpha)$
$\rightarrow Supp(\beta)\ UnSupp(\beta)\ \}$

In this example, a, b, c, α, β are all supported since they are prima-facie. Moreover a, c, (resp. β) are not attacked and so are e-accepted (resp. e-valid). α is attacked and not defended so it is not e-valid. And finally, since α is not e-valid and even if its source is e-accepted, b can be accepted and so it is e-accepted. In this example, there is only one complete, preferred and stable structure: $(\{a, b, c\}, \{\beta\}, \varnothing)$.

Prop 3 produces the following structures:

- Conflict-free structures: there are 25 among the 32 possible structures; are not conflict-free structures that contain together a, b and α, or those containing c, β, α.
- Admissible structures:

```
[]
[c]
[c, beta]
[beta]
[b, c, beta]
[a, beta]
[a]
[a, c]
[a, c, beta]
[a, b, c, beta]
```

- Preferred structures:

```
[a, b, c, beta]
```

- Grounded structure:

  `[a, b, c, beta]`

- Complete structures:

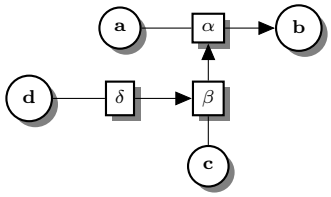
  `[a, b, c, beta]`

- Stable structures:

  `[a, b, c, beta]`

□

Example 5 (cont'd):



$\Sigma = \{$
$Supp(a)\ Acc(a) \rightarrow eAcc(a)$
$eAcc(a) \rightarrow Acc(a)$
$eAcc(a) \rightarrow Supp(a)$
$Supp(b)\ Acc(b) \rightarrow eAcc(b)$
$eAcc(b) \rightarrow Acc(b)$
$eAcc(b) \rightarrow Supp(b)$
$Supp(c)\ Acc(c) \rightarrow eAcc(c)$
$eAcc(c) \rightarrow Acc(c)$
$eAcc(c) \rightarrow Supp(c)$
$Supp(d)\ Acc(d) \rightarrow eAcc(d)$
$eAcc(d) \rightarrow Acc(d)$
$eAcc(d) \rightarrow Supp(d)$
$Supp(\alpha)\ Val(\alpha) \rightarrow eVal(\alpha)$
$eVal(\alpha) \rightarrow Val(\alpha)$
$eVal(\alpha) \rightarrow Supp(\alpha)$
$Supp(\beta)\ Val(\beta) \rightarrow eVal(\beta)$
$eVal(\beta) \rightarrow Val(\beta)$
$eVal(\beta) \rightarrow Supp(\beta)$
$Supp(\delta)\ Val(\delta) \rightarrow eVal(\delta)$
$eVal(\delta) \rightarrow Val(\delta)$
$eVal(\delta) \rightarrow Supp(\delta)$
$Val(\alpha)\ eVal(\beta)\ eAcc(c) \rightarrow$

$Val(\beta)\ eVal(\delta)\ eAcc(d) \rightarrow$
$eVal(\alpha)\ eAcc(a) \rightarrow NAcc(b)$
$NAcc(b)\ Acc(b) \rightarrow$
$\rightarrow Supp(a)$
$\rightarrow Supp(b)$
$\rightarrow Supp(c)$
$\rightarrow Supp(d)$
$\rightarrow Supp(\alpha)$
$\rightarrow Supp(\beta)$
$\rightarrow Supp(\delta)\ \}$

$\Sigma_{ss} = \Sigma \cup \{$
$UnSupp(a) \rightarrow$
$UnSupp(b) \rightarrow$
$UnSupp(c) \rightarrow$
$UnSupp(d) \rightarrow$
$UnSupp(\alpha) \rightarrow$
$UnSupp(\beta) \rightarrow$
$UnSupp(\delta) \rightarrow \}$

$\Sigma_d = \Sigma_{ss} \cup \{$
$Val(\beta) \rightarrow UnSupp(d)\ UnSupp(\delta)$
$Val(\alpha) \rightarrow eVal(\delta)\ UnSupp(c)\ UnSupp(\beta)$
$Val(\alpha) \rightarrow eAcc(d)\ UnSupp(c)\ UnSupp(\beta)$
$Acc(b) \rightarrow eVal(\beta)\ UnSupp(a)\ UnSupp(\alpha)$
$Acc(b) \rightarrow eAcc(c)\ UnSupp(a)\ UnSupp(\alpha)\ \}$

$\Sigma_r = \Sigma_{ss} \cup \{$
$\rightarrow Acc(a)$
$\rightarrow Acc(c)$
$\rightarrow Acc(d)$
$UnSupp(\alpha) \rightarrow Acc(b)$
$UnSupp(a) \rightarrow Acc(b)$
$eVal(\beta)\ eAcc(c) \rightarrow Acc(b)$
$\rightarrow Val(\delta)$
$UnSupp(\delta) \rightarrow Val(\beta)$
$UnSupp(d) \rightarrow Val(\beta)$
$UnSupp(\beta) \rightarrow Val(\alpha)$
$UnSupp(c) \rightarrow Val(\alpha)$
$eVal(\delta)\ eAcc(d) \rightarrow Val(\alpha)\ \}$

$\Sigma_s = \Sigma_{ss} \cup \{$
$\rightarrow Acc(a)$
$\rightarrow eVal(\alpha)\ Acc(b)$
$\rightarrow eAcc(a)\ Acc(b)$
$\rightarrow Acc(c)$
$\rightarrow Acc(d)$
$\rightarrow eVal(\beta)\ Val(\alpha)$
$\rightarrow eAcc(c)\ Val(\alpha)$
$\rightarrow eVal(\delta)\ Val(\beta)$

$\rightarrow eAcc(d)\ Val(\beta)$
$\rightarrow Val(\delta)$
$\rightarrow Supp(a)\ UnSupp(a)$
$\rightarrow Supp(b)\ UnSupp(b)$
$\rightarrow Supp(c)\ UnSupp(c)$
$\rightarrow Supp(d)\ UnSupp(d)$
$\rightarrow Supp(\alpha)\ UnSupp(\alpha)$
$\rightarrow Supp(\beta)\ UnSupp(\beta)$
$\rightarrow Supp(\delta)\ UnSupp(\delta)\ \}$

In this example, a, b, c, d, α, β, δ are all supported since they are prima-facie. Moreover a, c, d (resp. δ) are not attacked and so are e-accepted (resp. e-valid). β is attacked and not defended so it is e-valid. α is attacked but it is defended by an e-valid attack from an e-accepted argument, so it is e-valid. And finally, since α is e-valid and its source is e-accepted, b cannot be accepted and so it is not e-accepted. In this example, there is only one complete, preferred and stable structure: $(\{a, c, d\}, \{\alpha, \delta\}, \varnothing)$.

Prop 3 produces the following structures:

- Conflict-free structures: there are 89 among the 128 possible structures; are not conflict-free structures that contain together a, b and α, or those containing c, β, α, or those containing d, β, δ.
- Admissible structures:

```
[]
[c]
[c, d]
[d]
[c, d, delta]
[delta]
[c, delta]
[d, delta]
[d, alpha, delta]
[c, d, alpha, delta]
[a, c, d, alpha, delta]
[a]
[a, c]
[a, c, d]
[a, d]
[a, c, d, delta]
[a, delta]
[a, c, delta]
[a, d, delta]
[a, d, alpha, delta]
```

- Preferred structures:

```
[a, c, d, alpha, delta]
```

- Grounded structure:

```
[a, c, d, alpha, delta]
```

- Complete structures:

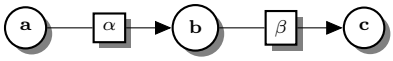
```
[a, c, d, alpha, delta]
```

- Stable structures:

```
[a, c, d, alpha, delta]
```

□

Example 10 (cont'd): Consider the AF represented by:



It can be encoded by the following simplified bases:

$\Sigma = \{$
$Supp(a)\ Acc(a) \rightarrow eAcc(a)$
$eAcc(a) \rightarrow Acc(a)$
$eAcc(a) \rightarrow Supp(a)$
$Supp(b)\ Acc(b) \rightarrow eAcc(b)$
$eAcc(b) \rightarrow Acc(b)$
$eAcc(b) \rightarrow Supp(b)$
$Supp(c)\ Acc(c) \rightarrow eAcc(c)$
$eAcc(c) \rightarrow Acc(c)$
$eAcc(c) \rightarrow Supp(c)$
$Supp(\alpha)\ Val(\alpha) \rightarrow eVal(\alpha)$
$eVal(\alpha) \rightarrow Val(\alpha)$
$eVal(\alpha) \rightarrow Supp(\alpha)$
$Supp(\beta)\ Val(\beta) \rightarrow eVal(\beta)$
$eVal(\beta) \rightarrow Val(\beta)$
$eVal(\beta) \rightarrow Supp(\beta)$
$eVal(\alpha)\ eAcc(a) \rightarrow NAcc(b)$
$eVal(\beta)\ eAcc(b) \rightarrow NAcc(c)$
$NAcc(b)\ Acc(b) \rightarrow$
$NAcc(c)\ Acc(c) \rightarrow$
$\rightarrow Supp(a)$
$\rightarrow Supp(b)$
$\rightarrow Supp(c)$
$\rightarrow Supp(\alpha)$
$\rightarrow Supp(\beta)\ \}$

$\Sigma_{ss} = \Sigma \cup \{$
$UnSupp(a) \rightarrow$
$UnSupp(b) \rightarrow$

$UnSupp(c) \rightarrow$
$UnSupp(\alpha) \rightarrow$
$UnSupp(\beta) \rightarrow \}$

$\Sigma_d = \Sigma_{ss} \cup \{$
$Acc(b) \rightarrow UnSupp(a)\ UnSupp(\alpha)$
$Acc(c) \rightarrow eVal(\alpha)\ UnSupp(b)\ UnSupp(\beta)$
$Acc(c) \rightarrow eAcc(a)\ UnSupp(b)\ UnSupp(\beta)\ \}$

$\Sigma_r = \Sigma_{ss} \cup \{$
$\rightarrow Acc(a)$
$UnSupp(\alpha) \rightarrow Acc(b)$
$UnSupp(a) \rightarrow Acc(b)$
$UnSupp(\beta) \rightarrow Acc(c)$
$UnSupp(b) \rightarrow Acc(c)$
$eVal(\alpha)\ eAcc(a) \rightarrow Acc(c)$
$\rightarrow Val(\alpha)$
$\rightarrow Val(\beta)\ \}$

$\Sigma_s = \Sigma_{ss} \cup \{$
$\rightarrow Acc(a)$
$\rightarrow eVal(\alpha)\ Acc(b)$
$\rightarrow eAcc(a)\ Acc(b)$
$\rightarrow eVal(\beta)\ Acc(c)$
$\rightarrow eAcc(b)\ Acc(c)$
$\rightarrow Val(\alpha)$
$\rightarrow Val(\beta)$
$\rightarrow Supp(a)\ UnSupp(a)$
$\rightarrow Supp(b)\ UnSupp(b)$
$\rightarrow Supp(c)\ UnSupp(c)$
$\rightarrow Supp(\alpha)\ UnSupp(\alpha)$
$\rightarrow Supp(\beta)\ UnSupp(\beta)\ \}$

Prop 3 produces the following structures:

- Conflict-free structures: there are 25 among the 32 possible structures; are not conflict-free structures that contain together a, b and α, or those containing b, c, β.
- Admissible structures:

```
[]
[alpha]
[alpha, beta]
[beta]
[a, beta]
[a]
[a, alpha]
[a, c, alpha]
[a, c, alpha, beta]
[a, alpha, beta]
```

- Preferred structures:

  `[a, c, alpha, beta]`

- Grounded structure:

  `[a, c, alpha, beta]`

- Complete structures:

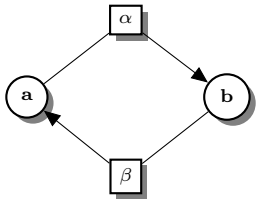
  `[a, c, alpha, beta]`

- Stable structures:

  `[a, c, alpha, beta]`

□

Example 20 *Consider the following RAF.*



$\Sigma = \{$
$Supp(a)\ Acc(a) \rightarrow eAcc(a)$
$eAcc(a) \rightarrow Acc(a)$
$eAcc(a) \rightarrow Supp(a)$
$Supp(b)\ Acc(b) \rightarrow eAcc(b)$
$eAcc(b) \rightarrow Acc(b)$
$eAcc(b) \rightarrow Supp(b)$
$Supp(\alpha)\ Val(\alpha) \rightarrow eVal(\alpha)$
$eVal(\alpha) \rightarrow Val(\alpha)$
$eVal(\alpha) \rightarrow Supp(\alpha)$
$Supp(\beta)\ Val(\beta) \rightarrow eVal(\beta)$
$eVal(\beta) \rightarrow Val(\beta)$
$eVal(\beta) \rightarrow Supp(\beta)$
$eVal(\alpha)\ eAcc(a) \rightarrow NAcc(b)$
$eVal(\beta)\ eAcc(b) \rightarrow NAcc(a)$
$NAcc(b)\ Acc(b) \rightarrow$
$NAcc(a)\ Acc(a) \rightarrow$
$\rightarrow Supp(a)$
$\rightarrow Supp(b)$

$\rightarrow Supp(\alpha)$
$\rightarrow Supp(\beta)$ }

$\Sigma_{ss} = \Sigma \cup \{$
$UnSupp(a) \rightarrow$
$UnSupp(b) \rightarrow$
$UnSupp(\alpha) \rightarrow$
$UnSupp(\beta) \rightarrow$ }

$\Sigma_d = \Sigma_{ss} \cup \{$
$Acc(a) \rightarrow eVal(\alpha)\ UnSupp(b)\ UnSupp(\beta)$
$Acc(a) \rightarrow eAcc(a)\ UnSupp(b)\ UnSupp(\beta)$
$Acc(b) \rightarrow eVal(\beta)\ UnSupp(a)\ UnSupp(\alpha)$
$Acc(b) \rightarrow eAcc(b)\ UnSupp(a)\ UnSupp(\alpha)$ }

$\Sigma_r = \Sigma_{ss} \cup \{$
$UnSupp(\beta) \rightarrow Acc(a)$
$UnSupp(b) \rightarrow Acc(a)$
$eVal(\alpha)\ eAcc(a) \rightarrow Acc(a)$
$UnSupp(\alpha) \rightarrow Acc(b)$
$UnSupp(a) \rightarrow Acc(b)$
$eVal(\beta)\ eAcc(b) \rightarrow Acc(b)$
$\rightarrow Val(\alpha)$
$\rightarrow Val(\beta)$ }

$\Sigma_s = \Sigma_{ss} \cup \{$
$\rightarrow eVal(\alpha)\ Acc(b)$
$\rightarrow eAcc(a)\ Acc(b)$
$\rightarrow eVal(\beta)\ Acc(a)$
$\rightarrow eAcc(b)\ Acc(a)$
$\rightarrow Val(\alpha)$
$\rightarrow Val(\beta)$
$\rightarrow Supp(a)\ UnSupp(a)$
$\rightarrow Supp(b)\ UnSupp(b)$
$\rightarrow Supp(\alpha)\ UnSupp(\alpha)$
$\rightarrow Supp(\beta)\ UnSupp(\beta)$ }

Prop 3 produces the following structures:

- *Conflict-free structures: there are 13 among the 16 possible structures; are not conflict-free structures that contain together a, b and α, or those containing a, b, β.*
- *Admissible structures:*

```
[]
[alpha]
[alpha, beta]
[beta]
[b, alpha, beta]
[b, beta]
```

```
[a, alpha, beta]
[a, alpha]
```

- *Preferred structures:*

```
[b, alpha, beta]
[a, alpha, beta]
```

- *Grounded structure:*

```
[alpha, beta]
```

- *Complete structures:*

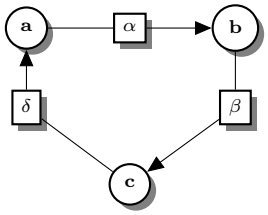
```
[alpha, beta]
[b, alpha, beta]
[a, alpha, beta]
```

- *Stable structures:*

```
[b, alpha, beta]
[a, alpha, beta]
```

□

Example 21 *Consider the following RAF.*



$\Sigma = \{$
$Supp(a)\ Acc(a) \rightarrow eAcc(a)$
$eAcc(a) \rightarrow Acc(a)$
$eAcc(a) \rightarrow Supp(a)$
$Supp(b)\ Acc(b) \rightarrow eAcc(b)$
$eAcc(b) \rightarrow Acc(b)$
$eAcc(b) \rightarrow Supp(b)$
$Supp(c)\ Acc(c) \rightarrow eAcc(c)$
$eAcc(c) \rightarrow Acc(c)$
$eAcc(c) \rightarrow Supp(c)$
$Supp(\alpha)\ Val(\alpha) \rightarrow eVal(\alpha)$
$eVal(\alpha) \rightarrow Val(\alpha)$

$eVal(\alpha) \rightarrow Supp(\alpha)$
$Supp(\beta)\ Val(\beta) \rightarrow eVal(\beta)$
$eVal(\beta) \rightarrow Val(\beta)$
$eVal(\beta) \rightarrow Supp(\beta)$
$Supp(\delta)\ Val(\delta) \rightarrow eVal(\delta)$
$eVal(\delta) \rightarrow Val(\delta)$
$eVal(\delta) \rightarrow Supp(\delta)$
$eVal(\alpha)\ eAcc(a) \rightarrow NAcc(b)$
$eVal(\beta)\ eAcc(b) \rightarrow NAcc(c)$
$eVal(\delta)\ eAcc(c) \rightarrow NAcc(a)$
$NAcc(b)\ Acc(b) \rightarrow$
$NAcc(c)\ Acc(c) \rightarrow$
$NAcc(a)\ Acc(a) \rightarrow$
$\rightarrow Supp(a)$
$\rightarrow Supp(b)$
$\rightarrow Supp(c)$
$\rightarrow Supp(\alpha)$
$\rightarrow Supp(\beta)$
$\rightarrow Supp(\delta)\ \}$

$\Sigma_{ss} = \Sigma \cup \{$
$UnSupp(a) \rightarrow$
$UnSupp(b) \rightarrow$
$UnSupp(c) \rightarrow$
$UnSupp(\alpha) \rightarrow$
$UnSupp(\beta) \rightarrow$
$UnSupp(\delta) \rightarrow \}$

$\Sigma_d = \Sigma_{ss} \cup \{$
$Acc(a) \rightarrow eVal(\beta)\ UnSupp(c)\ UnSupp(\delta)$
$Acc(a) \rightarrow eAcc(b)\ UnSupp(c)\ UnSupp(\delta)$
$Acc(b) \rightarrow eVal(\delta)\ UnSupp(a)\ UnSupp(\alpha)$
$Acc(b) \rightarrow eAcc(c)\ UnSupp(a)\ UnSupp(\alpha)$
$Acc(c) \rightarrow eVal(\alpha)\ UnSupp(b)\ UnSupp(\beta)$
$Acc(c) \rightarrow eAcc(a)\ UnSupp(b)\ UnSupp(\beta)\ \}$

$\Sigma_r = \Sigma_{ss} \cup \{$
$UnSupp(\delta) \rightarrow Acc(a)$
$UnSupp(c) \rightarrow Acc(a)$
$eVal(\beta)\ eAcc(b) \rightarrow Acc(a)$
$UnSupp(\alpha) \rightarrow Acc(b)$
$UnSupp(a) \rightarrow Acc(b)$
$eVal(\delta)\ eAcc(c) \rightarrow Acc(b)$
$UnSupp(\beta) \rightarrow Acc(c)$
$UnSupp(b) \rightarrow Acc(c)$
$eVal(\alpha)\ eAcc(a) \rightarrow Acc(c)$
$\rightarrow Val(\alpha)$
$\rightarrow Val(\beta)$
$\rightarrow Val(\delta)\ \}$

$\Sigma_s = \Sigma_{ss} \cup \{$
$\rightarrow eVal(\alpha)\ Acc(b)$
$\rightarrow eAcc(a)\ Acc(b)$
$\rightarrow eVal(\delta)\ Acc(a)$
$\rightarrow eAcc(c)\ Acc(a)$
$\rightarrow eVal(\beta)\ Acc(c)$
$\rightarrow eAcc(b)\ Acc(c)$
$\rightarrow Val(\alpha)$
$\rightarrow Val(\beta)$
$\rightarrow Val(\delta)$
$\rightarrow Supp(a)\ UnSupp(a)$
$\rightarrow Supp(b)\ UnSupp(b)$
$\rightarrow Supp(c)\ UnSupp(c)$
$\rightarrow Supp(\alpha)\ UnSupp(\alpha)$
$\rightarrow Supp(\beta)\ UnSupp(\beta)$
$\rightarrow Supp(\delta)\ UnSupp(\delta)\ \}$

Prop 3 produces the following structures:

- *Conflict-free structures: there are 45 among the 64 possible structures; are not conflict-free structures that contain together a, b and α, or those containing b, c and β, or those containing a, c and δ.*
- *Admissible structures:*

```
[]
[alpha]
[alpha, beta]
[beta]
[alpha, beta, delta]
[delta]
[alpha, delta]
[beta, delta]
```

- *Preferred structures:*

```
[alpha, beta, delta]
```

- *Grounded structure:*

```
[alpha, beta, delta]
```

- *Complete structures:*

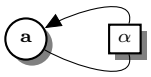
```
[alpha, beta, delta]
```

- *Stable structures:*

```
none
```

□

Example 22 *Consider the following RAF.*



$\Sigma = \{$
$Supp(a)\ Acc(a) \rightarrow eAcc(a)$
$eAcc(a) \rightarrow Acc(a)$
$eAcc(a) \rightarrow Supp(a)$
$Supp(\alpha)\ Val(\alpha) \rightarrow eVal(\alpha)$
$eVal(\alpha) \rightarrow Val(\alpha)$
$eVal(\alpha) \rightarrow Supp(\alpha)$
$eVal(\alpha)\ eAcc(a) \rightarrow NAcc(a)$
$NAcc(a)\ Acc(a) \rightarrow$
$\rightarrow Supp(a)$
$\rightarrow Supp(\alpha)\ \}$
$\Sigma_{ss} = \Sigma \cup \{$
$UnSupp(a) \rightarrow$
$UnSupp(\alpha) \rightarrow \}$
$\Sigma_d = \Sigma_{ss} \cup \{$
$Acc(a) \rightarrow eVal(\alpha)\ UnSupp(a)\ UnSupp(\alpha)$
$Acc(a) \rightarrow eAcc(a)\ UnSupp(a)\ UnSupp(\alpha)\ \}$
$\Sigma_r = \Sigma_{ss} \cup \{$
$UnSupp(\alpha) \rightarrow Acc(a)$
$UnSupp(a) \rightarrow Acc(a)$
$eVal(\alpha)\ eAcc(a) \rightarrow Acc(a)$
$\rightarrow Val(\alpha)\ \}$
$\Sigma_s = \Sigma_{ss} \cup \{$
$\rightarrow eVal(\alpha)\ Acc(a)$
$\rightarrow eAcc(a)\ Acc(a)$
$\rightarrow Val(\alpha)$
$\rightarrow Supp(a)\ UnSupp(a)$
$\rightarrow Supp(\alpha)\ UnSupp(\alpha)\ \}$

Prop 3 produces the following structures:

- *Conflict-free structures: there are 3 among the 4 possible structures; is not conflict-free the structure that contains together a and α.*

```
[]
[alpha]
[a]
```

- *Admissible structures:*

```
[]
[alpha]
```

- *Preferred structures:*

```
[alpha]
```

- *Grounded structure:*

```
[alpha]
```

- *Complete structures:*

```
[alpha]
```

- *Stable structures:*

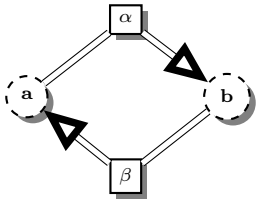
```
none
```

□

B.3 Examples of REBAF with cycles of support

In this section, due to the presence of support cycles, we can see that Prop 3 produces structures that do not respect the admissibility property. This problem is solved using Proposition 4.

Example 13 (cont'd):
Consider the following REBAF.



$\Sigma = \{$
$Supp(a)\ Acc(a) \rightarrow eAcc(a)$
$eAcc(a) \rightarrow Acc(a)$
$eAcc(a) \rightarrow Supp(a)$
$Supp(b)\ Acc(b) \rightarrow eAcc(b)$
$eAcc(b) \rightarrow Acc(b)$
$eAcc(b) \rightarrow Supp(b)$

$Supp(\alpha)\ Val(\alpha) \rightarrow eVal(\alpha)$
$eVal(\alpha) \rightarrow Val(\alpha)$
$eVal(\alpha) \rightarrow Supp(\alpha)$
$Supp(\beta)\ Val(\beta) \rightarrow eVal(\beta)$
$eVal(\beta) \rightarrow Val(\beta)$
$eVal(\beta) \rightarrow Supp(\beta)$
$eVal(\alpha)\ eAcc(a) \rightarrow Supp(b)$
$eVal(\beta)\ eAcc(b) \rightarrow Supp(a)$
$\rightarrow Supp(\alpha)$
$\rightarrow Supp(\beta)\ \}$

$\Sigma_{ss} = \Sigma \cup \{$
$Supp(b) \rightarrow eVal(\alpha)$
$Supp(b) \rightarrow eAcc(a)$
$Supp(a) \rightarrow eVal(\beta)$
$Supp(a) \rightarrow eAcc(b)$
$UnSupp(\alpha) \rightarrow$
$UnSupp(\beta) \rightarrow$
$UnSupp(a) \rightarrow UnSupp(b)\ UnSupp(\beta)$
$UnSupp(\beta) \rightarrow UnSupp(a)$
$UnSupp(b) \rightarrow UnSupp(a)$
$UnSupp(b) \rightarrow UnSupp(a)\ UnSupp(\alpha)$
$UnSupp(\alpha) \rightarrow UnSupp(b)$
$UnSupp(a) \rightarrow UnSupp(b)\ \}$

$\Sigma_d = \Sigma_{ss} \cup \varnothing$

$\Sigma_r = \Sigma_{ss} \cup \{$
$\rightarrow Acc(a)$
$\rightarrow Acc(b)$
$\rightarrow Val(\alpha)$
$\rightarrow Val(\beta)\ \}$

$\Sigma_s = \Sigma_{ss} \cup \{$
$\rightarrow Acc(b)$
$\rightarrow Acc(a)$
$\rightarrow Val(\alpha)$
$\rightarrow Val(\beta)$
$\rightarrow Supp(a)\ UnSupp(a)$
$\rightarrow Supp(b)\ UnSupp(b)$
$\rightarrow Supp(\alpha)\ UnSupp(\alpha)$
$\rightarrow Supp(\beta)\ UnSupp(\beta)\ \}$

Prop 3 produces the following structures:

- Conflict-free structures: all structures are conflict-free.
- Admissible structures:

```
[]
[alpha]
```

```
[alpha, beta]
[beta]
[a, b, alpha, beta]
```

Nevertheless, this last structure is not admissible. And so the following results will not be correct.

- Preferred structures:

```
[a, b, alpha, beta]
```

- Grounded structure:

```
[alpha, beta]
```

- Complete structures:

```
[alpha, beta]
[a, b, alpha, beta]
```

- Stable structures:

```
[alpha, beta]
[a, b, alpha, beta]
```

Prop 4 produces the following admissible structures:

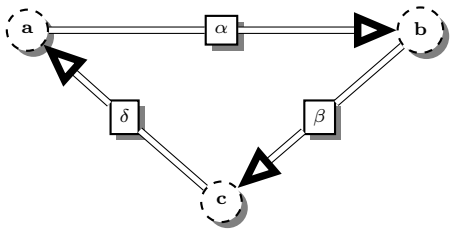
```
[]
[alpha]
[alpha, beta]
[beta]
```

And the complete, grounded and preferred structure is:

```
[alpha, beta]
```

□

Example 14 (cont'd):
Consider the following REBAF.



$\Sigma = \{$
$Supp(a)\ Acc(a) \rightarrow eAcc(a)$
$eAcc(a) \rightarrow Acc(a)$
$eAcc(a) \rightarrow Supp(a)$
$Supp(b)\ Acc(b) \rightarrow eAcc(b)$
$eAcc(b) \rightarrow Acc(b)$
$eAcc(b) \rightarrow Supp(b)$
$Supp(c)\ Acc(c) \rightarrow eAcc(c)$
$eAcc(c) \rightarrow Acc(c)$
$eAcc(c) \rightarrow Supp(c)$
$Supp(\alpha)\ Val(\alpha) \rightarrow eVal(\alpha)$
$eVal(\alpha) \rightarrow Val(\alpha)$
$eVal(\alpha) \rightarrow Supp(\alpha)$
$Supp(\beta)\ Val(\beta) \rightarrow eVal(\beta)$
$eVal(\beta) \rightarrow Val(\beta)$
$eVal(\beta) \rightarrow Supp(\beta)$
$Supp(\delta)\ Val(\delta) \rightarrow eVal(\delta)$
$eVal(\delta) \rightarrow Val(\delta)$
$eVal(\delta) \rightarrow Supp(\delta)$
$eVal(\alpha)\ eAcc(a) \rightarrow Supp(b)$
$eVal(\beta)\ eAcc(b) \rightarrow Supp(c)$
$eVal(\delta)\ eAcc(c) \rightarrow Supp(a)$
$\rightarrow Supp(\alpha)$
$\rightarrow Supp(\beta)$
$\rightarrow Supp(\delta)\ \}$
$\Sigma_{ss} = \Sigma \cup \{$
$Supp(b) \rightarrow eVal(\alpha)$
$Supp(b) \rightarrow eAcc(a)$
$Supp(c) \rightarrow eVal(\beta)$
$Supp(c) \rightarrow eAcc(b)$
$Supp(a) \rightarrow eVal(\delta)$
$Supp(a) \rightarrow eAcc(c)$
$UnSupp(\alpha) \rightarrow$
$UnSupp(\beta) \rightarrow$
$UnSupp(\delta) \rightarrow$
$UnSupp(a) \rightarrow UnSupp(c)\ UnSupp(\delta)$
$UnSupp(\delta) \rightarrow UnSupp(a)$
$UnSupp(c) \rightarrow UnSupp(a)$
$UnSupp(b) \rightarrow UnSupp(a)\ UnSupp(\alpha)$
$UnSupp(\alpha) \rightarrow UnSupp(b)$
$UnSupp(a) \rightarrow UnSupp(b)$
$UnSupp(c) \rightarrow UnSupp(b)\ UnSupp(\beta)$
$UnSupp(\beta) \rightarrow UnSupp(c)$
$UnSupp(b) \rightarrow UnSupp(c)\ \}$
$\Sigma_d = \Sigma_{ss} \cup \varnothing$
$\Sigma_r = \Sigma_{ss} \cup \{$

$\rightarrow Acc(a)$
$\rightarrow Acc(b)$
$\rightarrow Acc(c)$
$\rightarrow Val(\alpha)$
$\rightarrow Val(\beta)$
$\rightarrow Val(\delta)$ }
$\Sigma_s = \Sigma_{ss} \cup \{$
$\rightarrow Acc(b)$
$\rightarrow Acc(a)$
$\rightarrow Acc(c)$
$\rightarrow Val(\alpha)$
$\rightarrow Val(\beta)$
$\rightarrow Val(\delta)$
$\rightarrow Supp(a)\ UnSupp(a)$
$\rightarrow Supp(b)\ UnSupp(b)$
$\rightarrow Supp(c)\ UnSupp(c)$
$\rightarrow Supp(\alpha)\ UnSupp(\alpha)$
$\rightarrow Supp(\beta)\ UnSupp(\beta)$
$\rightarrow Supp(\delta)\ UnSupp(\delta)$ }

Prop 3 produces the following structures:

- Conflict-free structures: all structures are conflict-free.
- Admissible structures:

```
[]
[alpha]
[alpha, beta]
[beta]
[alpha, beta, delta]
[delta]
[alpha, delta]
[beta, delta]
[a, b, c, alpha, beta, delta]
```

  Nevertheless, this last structure is not admissible. And so the following results will not be correct.

- Preferred structures:

```
[a, b, c, alpha, beta, delta]
```

- Grounded structure:

```
[alpha, beta, delta]
```

- Complete structures:

```
[alpha, beta, delta]
[a, b, c, alpha, beta, delta]
```

- Stable structures:

```
[alpha, beta, delta]
[a, b, c, alpha, beta, delta]
```

Prop 4 produces the following admissible structures:

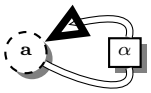
```
[]
[alpha]
[alpha, beta]
[beta]
[alpha, beta, delta]
[delta]
[alpha, delta]
[beta, delta]
```

And the complete, grounded and preferred structure is:

```
[alpha, beta, delta]
```

□

Example 15 (cont'd):
Consider the following REBAF.



$\Sigma = \{$
$Supp(a)\ Acc(a) \rightarrow eAcc(a)$
$eAcc(a) \rightarrow Acc(a)$
$eAcc(a) \rightarrow Supp(a)$
$Supp(\alpha)\ Val(\alpha) \rightarrow eVal(\alpha)$
$eVal(\alpha) \rightarrow Val(\alpha)$
$eVal(\alpha) \rightarrow Supp(\alpha)$
$eVal(\alpha)\ eAcc(a) \rightarrow Supp(a)$
$\rightarrow Supp(\alpha)\ \}$
$\Sigma_{ss} = \Sigma \cup \{$
$Supp(a) \rightarrow eVal(\alpha)$
$Supp(a) \rightarrow eAcc(a)$
$UnSupp(\alpha) \rightarrow$
$UnSupp(a) \rightarrow UnSupp(a)\ UnSupp(\alpha)$
$UnSupp(\alpha) \rightarrow UnSupp(a)$
$UnSupp(a) \rightarrow UnSupp(a)\ \}$

$\Sigma_d = \Sigma_{ss} \cup \varnothing$
$\Sigma_r = \Sigma_{ss} \cup \{$
$\quad \rightarrow Acc(a)$
$\quad \rightarrow Val(\alpha)\ \}$
$\Sigma_s = \Sigma_{ss} \cup \{$
$\quad \rightarrow Acc(a)$
$\quad \rightarrow Val(\alpha)$
$\quad \rightarrow Supp(a)\ UnSupp(a)$
$\quad \rightarrow Supp(\alpha)\ UnSupp(\alpha)\ \}$

Prop 3 produces the following structures:

- Conflict-free structures: all structures are conflict-free.
- Admissible structures:

```
[]
[alpha]
[a, alpha]
```

  Nevertheless, this last structure is not admissible. And so the following results will not be correct.

- Preferred structures:

```
[a, alpha]
```

- Grounded structure:

```
[alpha]
```

- Complete structures:

```
[alpha]
[a, alpha]
```

- Stable structures:

```
[alpha]
[a, alpha]
```

Prop 4 produces the following admissible structures:

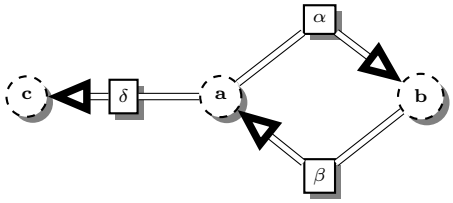
```
[]
[alpha]
```

And the complete, grounded and preferred structure is:

```
[alpha]
```

□

Example 23 *Consider the following REBAF.*



$\Sigma = \{$
$Supp(a)\ Acc(a) \rightarrow eAcc(a)$
$eAcc(a) \rightarrow Acc(a)$
$eAcc(a) \rightarrow Supp(a)$
$Supp(b)\ Acc(b) \rightarrow eAcc(b)$
$eAcc(b) \rightarrow Acc(b)$
$eAcc(b) \rightarrow Supp(b)$
$Supp(c)\ Acc(c) \rightarrow eAcc(c)$
$eAcc(c) \rightarrow Acc(c)$
$eAcc(c) \rightarrow Supp(c)$
$Supp(\alpha)\ Val(\alpha) \rightarrow eVal(\alpha)$
$eVal(\alpha) \rightarrow Val(\alpha)$
$eVal(\alpha) \rightarrow Supp(\alpha)$
$Supp(\beta)\ Val(\beta) \rightarrow eVal(\beta)$
$eVal(\beta) \rightarrow Val(\beta)$
$eVal(\beta) \rightarrow Supp(\beta)$
$Supp(\delta)\ Val(\delta) \rightarrow eVal(\delta)$
$eVal(\delta) \rightarrow Val(\delta)$
$eVal(\delta) \rightarrow Supp(\delta)$
$eVal(\alpha)\ eAcc(a) \rightarrow Supp(b)$
$eVal(\beta)\ eAcc(b) \rightarrow Supp(a)$
$eVal(\delta)\ eAcc(a) \rightarrow Supp(c)$
$\rightarrow Supp(\alpha)$
$\rightarrow Supp(\beta)$
$\rightarrow Supp(\delta)\ \}$
$\Sigma_{ss} = \Sigma \cup \{$
$Supp(b) \rightarrow eVal(\alpha)$
$Supp(b) \rightarrow eAcc(a)$
$Supp(a) \rightarrow eVal(\beta)$
$Supp(a) \rightarrow eAcc(b)$
$Supp(c) \rightarrow eVal(\alpha)$
$Supp(c) \rightarrow eAcc(a)$
$UnSupp(\alpha) \rightarrow$
$UnSupp(\beta) \rightarrow$
$UnSupp(\delta) \rightarrow$

$UnSupp(a) \rightarrow UnSupp(b)\ UnSupp(\beta)$
$UnSupp(\beta) \rightarrow UnSupp(a)$
$UnSupp(b) \rightarrow UnSupp(a)$
$UnSupp(b) \rightarrow UnSupp(a)\ UnSupp(\alpha)$
$UnSupp(\alpha) \rightarrow UnSupp(b)$
$UnSupp(a) \rightarrow UnSupp(b)$
$UnSupp(c) \rightarrow UnSupp(a)\ UnSupp(\alpha)$
$UnSupp(\alpha) \rightarrow UnSupp(c)$
$UnSupp(a) \rightarrow UnSupp(c)\ \}$

$\Sigma_d = \Sigma_{ss} \cup \varnothing$

$\Sigma_r = \Sigma_{ss} \cup \{$
$\rightarrow Acc(a)$
$\rightarrow Acc(b)$
$\rightarrow Acc(c)$
$\rightarrow Val(\alpha)$
$\rightarrow Val(\beta)$
$\rightarrow Val(\delta)\ \}$

$\Sigma_s = \Sigma_{ss} \cup \{$
$\rightarrow Acc(b)$
$\rightarrow Acc(a)$
$\rightarrow Acc(c)$
$\rightarrow Val(\alpha)$
$\rightarrow Val(\beta)$
$\rightarrow Val(\delta)$
$\rightarrow Supp(a)\ UnSupp(a)$
$\rightarrow Supp(b)\ UnSupp(b)$
$\rightarrow Supp(c)\ UnSupp(c)$
$\rightarrow Supp(\alpha)\ UnSupp(\alpha)$
$\rightarrow Supp(\beta)\ UnSupp(\beta)$
$\rightarrow Supp(\delta)\ UnSupp(\delta)\ \}$

Prop 3 produces the following structures:

- *Conflict-free structures: all structures are conflict-free.*
- *Admissible structures:*

```
[ ]
[ alpha ]
[ alpha, beta ]
[ beta ]
[ alpha, beta, delta ]
[ delta ]
[ alpha, delta ]
[ beta, delta ]
[ a, b, alpha, beta ]
[ a, b, c, alpha, beta ]
```

```
[ a, b, c, alpha, beta, delta ]
[ a, b, alpha, beta, delta ]
```

Nevertheless, the four last structures are not admissible. And so the following results will not be correct.

- *Preferred structures:*

```
[ a, b, c, alpha, beta, delta ]
```

- *Grounded structure:*

```
[ alpha, beta, delta ]
```

- *Complete structures:*

```
[ alpha, beta, delta ]
[ a, b, c, alpha, beta, delta ]
```

- *Stable structures:*

```
[ alpha, beta, delta ]
[ a, b, c, alpha, beta, delta ]
```

Prop 4 produces the following admissible structures:

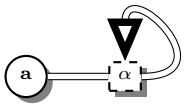
```
[ ]
[ alpha ]
[ alpha, beta ]
[ beta ]
[ alpha, beta, delta ]
[ delta ]
[ alpha, delta ]
[ beta, delta ]
```

And the complete, grounded and preferred structure is:

```
[alpha, beta, delta]
```

□

Example 24 *Consider the following REBAF.*



$\Sigma = \{$
$Supp(a)\ Acc(a) \rightarrow eAcc(a)$
$eAcc(a) \rightarrow Acc(a)$
$eAcc(a) \rightarrow Supp(a)$
$Supp(\alpha)\ Val(\alpha) \rightarrow eVal(\alpha)$
$eVal(\alpha) \rightarrow Val(\alpha)$
$eVal(\alpha) \rightarrow Supp(\alpha)$
$eVal(\alpha)\ eAcc(a) \rightarrow Supp(\alpha)$
$\rightarrow Supp(a)\ \}$
$\Sigma_{ss} = \Sigma \cup \{$
$Supp(\alpha) \rightarrow eVal(\alpha)$
$Supp(\alpha) \rightarrow eAcc(a)$
$UnSupp(a) \rightarrow$
$UnSupp(\alpha) \rightarrow UnSupp(a)\ UnSupp(\alpha)$
$UnSupp(\alpha) \rightarrow UnSupp(\alpha)$
$UnSupp(a) \rightarrow UnSupp(\alpha)\ \}$
$\Sigma_d = \Sigma_{ss} \cup \varnothing$
$\Sigma_r = \Sigma_{ss} \cup \{$
$\rightarrow Acc(a)$
$\rightarrow Val(\alpha)\ \}$
$\Sigma_s = \Sigma_{ss} \cup \{$
$\rightarrow Acc(a)$
$\rightarrow Val(\alpha)$
$\rightarrow Supp(a)\ UnSupp(a)$
$\rightarrow Supp(\alpha)\ UnSupp(\alpha)\ \}$

Prop 3 produces the following structures:

- *Conflict-free structures: all structures are conflict-free.*
- *Admissible structures:*

  ```
  []
  [a]
  [a, alpha]
  ```

  Nevertheless, this last structure is not admissible. And so the following results will not be correct.

- *Preferred structures:*

  ```
  [a, alpha]
  ```

- *Grounded structure:*

  ```
  [a]
  ```

- *Complete structures:*

```
[a]
[a, alpha]
```

- *Stable structures:*

```
[a]
[a, alpha]
```

Prop 4 produces the following admissible structures:

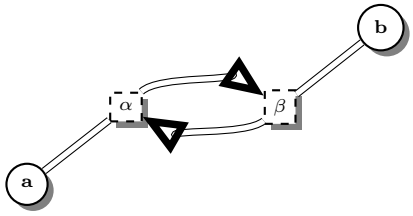
```
[]
[a]
```

And the complete, grounded and preferred structure is:

```
[a]
```

□

Example 25 *Consider the following REBAF.*



$\Sigma = \{$
$Supp(a)\ Acc(a) \rightarrow eAcc(a)$
$eAcc(a) \rightarrow Acc(a)$
$eAcc(a) \rightarrow Supp(a)$
$Supp(b)\ Acc(b) \rightarrow eAcc(b)$
$eAcc(b) \rightarrow Acc(b)$
$eAcc(b) \rightarrow Supp(b)$
$Supp(\alpha)\ Val(\alpha) \rightarrow eVal(\alpha)$
$eVal(\alpha) \rightarrow Val(\alpha)$
$eVal(\alpha) \rightarrow Supp(\alpha)$
$Supp(\beta)\ Val(\beta) \rightarrow eVal(\beta)$
$eVal(\beta) \rightarrow Val(\beta)$
$eVal(\beta) \rightarrow Supp(\beta)$
$eVal(\alpha)\ eAcc(a) \rightarrow Supp(\beta)$
$eVal(\beta)\ eAcc(b) \rightarrow Supp(\alpha)$
$\rightarrow Supp(a)$
$\rightarrow Supp(b)\ \}$
$\Sigma_{ss} = \Sigma \cup \{$
$Supp(\beta) \rightarrow eVal(\alpha)$

$Supp(\beta) \rightarrow eAcc(a)$
$Supp(\alpha) \rightarrow eVal(\beta)$
$Supp(\alpha) \rightarrow eAcc(b)$
$UnSupp(a) \rightarrow$
$UnSupp(b) \rightarrow$
$UnSupp(\alpha) \rightarrow UnSupp(b)\ UnSupp(\beta)$
$UnSupp(\beta) \rightarrow UnSupp(\alpha)$
$UnSupp(b) \rightarrow UnSupp(\alpha)$
$UnSupp(\beta) \rightarrow UnSupp(a)\ UnSupp(\alpha)$
$UnSupp(\alpha) \rightarrow UnSupp(\beta)$
$UnSupp(a) \rightarrow UnSupp(\beta)$ }
$\Sigma_d = \Sigma_{ss} \cup \varnothing$
$\Sigma_r = \Sigma_{ss} \cup \{$
$\rightarrow Acc(a)$
$\rightarrow Acc(b)$
$\rightarrow Val(\alpha)$
$\rightarrow Val(\beta)$ }
$\Sigma_s = \Sigma_{ss} \cup \{$
$\rightarrow Acc(b)$
$\rightarrow Acc(a)$
$\rightarrow Val(\alpha)$
$\rightarrow Val(\beta)$
$\rightarrow Supp(a)\ UnSupp(a)$
$\rightarrow Supp(b)\ UnSupp(b)$
$\rightarrow Supp(\alpha)\ UnSupp(\alpha)$
$\rightarrow Supp(\beta)\ UnSupp(\beta)$ }

Prop 3 produces the following structures:

- *Conflict-free structures: all structures are conflict-free.*
- *Admissible structures:*

```
[]
[b]
[a]
[a, b]
[a, b, alpha, beta]
```

  Nevertheless, this last structure is not admissible. And so the following results will not be correct.

- *Preferred structures:*

```
[a, b, alpha, beta]
```

- *Grounded structure:*

```
[a, b]
```

- *Complete structures:*

```
[a, b]
[a, b, alpha, beta]
```

- *Stable structures:*

```
[a, b]
[a, b, alpha, beta]
```

Prop 4 produces the following admissible structures:

```
[]
[b]
[a]
[a, b]
```

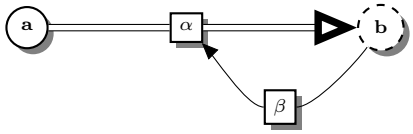
And the complete, grounded and preferred structure is:

```
[a, b]
```

□

B.4 Examples of REBAF with mixed cycles (support-attack)

Example 26 *Consider the following REBAF.*



$\Sigma = \{$
$Supp(a)\ Acc(a) \rightarrow eAcc(a)$
$eAcc(a) \rightarrow Acc(a)$
$eAcc(a) \rightarrow Supp(a)$
$Supp(b)\ Acc(b) \rightarrow eAcc(b)$
$eAcc(b) \rightarrow Acc(b)$
$eAcc(b) \rightarrow Supp(b)$
$Supp(\alpha)\ Val(\alpha) \rightarrow eVal(\alpha)$
$eVal(\alpha) \rightarrow Val(\alpha)$
$eVal(\alpha) \rightarrow Supp(\alpha)$
$Supp(\beta)\ Val(\beta) \rightarrow eVal(\beta)$

$eVal(\beta) \rightarrow Val(\beta)$
$eVal(\beta) \rightarrow Supp(\beta)$
$Val(\alpha)\ eVal(\beta)\ eAcc(b) \rightarrow$
$eVal(\alpha)\ eAcc(a) \rightarrow Supp(b)$
$\rightarrow Supp(a)$
$\rightarrow Supp(\alpha)$
$\rightarrow Supp(\beta)\ \}$

$\Sigma_{ss} = \Sigma \cup \{$
$Supp(b) \rightarrow eVal(\alpha)$
$Supp(b) \rightarrow eAcc(a)$
$UnSupp(a) \rightarrow$
$UnSupp(\alpha) \rightarrow$
$UnSupp(\beta) \rightarrow$
$UnSupp(b) \rightarrow eVal(\beta)\ UnSupp(a)\ UnSupp(\alpha)$
$UnSupp(b) \rightarrow eAcc(b)\ UnSupp(a)\ UnSupp(\alpha)$
$UnSupp(\alpha) \rightarrow UnSupp(b)$
$UnSupp(a) \rightarrow UnSupp(b)$
$eVal(\beta)\ eAcc(b) \rightarrow UnSupp(b)\ \}$

$\Sigma_d = \Sigma_{ss} \cup \{$
$Val(\alpha) \rightarrow UnSupp(b)\ UnSupp(\beta)\ \}$

$\Sigma_r = \Sigma_{ss} \cup \{$
$\rightarrow Acc(a)$
$\rightarrow Acc(b)$
$UnSupp(\beta) \rightarrow Val(\alpha)$
$UnSupp(b) \rightarrow Val(\alpha)$
$\rightarrow Val(\beta)\ \}$

$\Sigma_s = \Sigma_{ss} \cup \{$
$\rightarrow Acc(b)$
$\rightarrow Acc(a)$
$\rightarrow eVal(\beta)\ Val(\alpha)$
$\rightarrow eAcc(b)\ Val(\alpha)$
$\rightarrow Val(\beta)$
$\rightarrow Supp(a)\ UnSupp(a)$
$\rightarrow Supp(b)\ UnSupp(b)$
$\rightarrow Supp(\alpha)\ UnSupp(\alpha)$
$\rightarrow Supp(\beta)\ UnSupp(\beta)\ \}$

Here α belongs neither to $Def(U)$ (since b, the source of the attack to α, does not belong to an admissible structure) nor to $UnSupp(U)$ (since α is prima-facie), for any admissible structure U. That explains why there is no stable structure.

Prop 3 produces the following structures:

- *Conflict-free structures:there are 14 conflict-free structures among the 16 possible structures (are not conflict-free those that contain together b, α and β).*
- *Admissible structures:*

[ ]

```
[beta]
[a, beta]
[a]
```

- *Preferred structures:*

```
[a, beta]
```

- *Grounded structure:*

```
[a, beta]
```

- *Complete structures:*

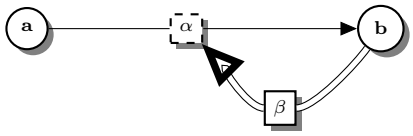
```
[a, beta]
```

- *Stable structures:*

```
none
```

□

Example 27 *Consider the following REBAF.*



$\Sigma = \{$
$Supp(a)\ Acc(a) \rightarrow eAcc(a)$
$eAcc(a) \rightarrow Acc(a)$
$eAcc(a) \rightarrow Supp(a)$
$Supp(b)\ Acc(b) \rightarrow eAcc(b)$
$eAcc(b) \rightarrow Acc(b)$
$eAcc(b) \rightarrow Supp(b)$
$Supp(\alpha)\ Val(\alpha) \rightarrow eVal(\alpha)$
$eVal(\alpha) \rightarrow Val(\alpha)$
$eVal(\alpha) \rightarrow Supp(\alpha)$
$Supp(\beta)\ Val(\beta) \rightarrow eVal(\beta)$
$eVal(\beta) \rightarrow Val(\beta)$
$eVal(\beta) \rightarrow Supp(\beta)$
$eVal(\alpha)\ eAcc(a) \rightarrow NAcc(b)$
$NAcc(b)\ Acc(b) \rightarrow$
$\rightarrow Supp(b)$
$\rightarrow Supp(a)$

$eVal(\beta)\ eAcc(b) \rightarrow Supp(\alpha)$
$\rightarrow Supp(\beta)\ \}$
$\Sigma_{ss} = \Sigma \cup \{$
$Supp(\alpha) \rightarrow eVal(\beta)$
$Supp(\alpha) \rightarrow eAcc(b)$
$UnSupp(a) \rightarrow$
$UnSupp(b) \rightarrow$
$UnSupp(\beta) \rightarrow$
$UnSupp(\alpha) \rightarrow UnSupp(b)\ UnSupp(\beta)$
$UnSupp(\beta) \rightarrow UnSupp(\alpha)$
$UnSupp(b) \rightarrow UnSupp(\alpha)\ \}$
$\Sigma_d = \Sigma_{ss} \cup \{$
$Acc(b) \rightarrow UnSupp(a)\ UnSupp(\alpha)\ \}$
$\Sigma_r = \Sigma_{ss} \cup \{$
$\rightarrow Acc(a)$
$UnSupp(\alpha) \rightarrow Acc(b)$
$UnSupp(a) \rightarrow Acc(b)$
$\rightarrow Val(\alpha)$
$\rightarrow Val(\beta)\ \}$
$\Sigma_s = \Sigma_{ss} \cup \{$
$\rightarrow Acc(a)$
$\rightarrow eVal(\alpha)\ Acc(b)$
$\rightarrow eAcc(a)\ Acc(b)$
$\rightarrow Val(\alpha)$
$\rightarrow Val(\beta)$
$\rightarrow Supp(a)\ UnSupp(a)$
$\rightarrow Supp(b)\ UnSupp(b)$
$\rightarrow Supp(\alpha)\ UnSupp(\alpha)$
$\rightarrow Supp(\beta)\ UnSupp(\beta)\ \}$

Here b belongs neither to $Def(U)$ (since α, the source of the attack to b, does not belong to an admissible structure) nor to $UnSupp(U)$ (since b is prima-facie), for any admissible structure U. That explains why there is no stable structure.

Prop 3 produces the following structures:

- *Conflict-free structures: there are 14 conflict-free structures among the 16 possible structures (are not conflict-free those that contain together a, b and α).*
- *Admissible structures:*

```
[]
[beta]
[a, beta]
[a]
```

- *Preferred structures:*

```
[a, beta]
```

- *Grounded structure:*

  `[a, beta]`

- *Complete structures:*

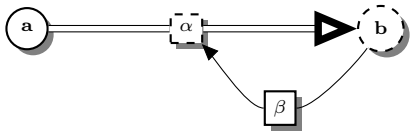
  `[a, beta]`

- *Stable structures:*

  `none`

□

Example 28 *Consider the following REBAF.*



$\Sigma = \{$
$Supp(a)\ Acc(a) \rightarrow eAcc(a)$
$eAcc(a) \rightarrow Acc(a)$
$eAcc(a) \rightarrow Supp(a)$
$Supp(b)\ Acc(b) \rightarrow eAcc(b)$
$eAcc(b) \rightarrow Acc(b)$
$eAcc(b) \rightarrow Supp(b)$
$Supp(\alpha)\ Val(\alpha) \rightarrow eVal(\alpha)$
$eVal(\alpha) \rightarrow Val(\alpha)$
$eVal(\alpha) \rightarrow Supp(\alpha)$
$Supp(\beta)\ Val(\beta) \rightarrow eVal(\beta)$
$eVal(\beta) \rightarrow Val(\beta)$
$eVal(\beta) \rightarrow Supp(\beta)$
$Val(\alpha)\ eVal(\beta)\ eAcc(b) \rightarrow$
$eVal(\alpha)\ eAcc(a) \rightarrow Supp(b)$
$\rightarrow Supp(a)$
$\rightarrow Supp(\beta)\ \}$
$\Sigma_{ss} = \Sigma \cup \{$
$Supp(b) \rightarrow eVal(\alpha)$
$Supp(b) \rightarrow eAcc(a)$
$Supp(\alpha) \rightarrow$
$UnSupp(a) \rightarrow$
$\rightarrow UnSupp(\alpha)$
$UnSupp(\beta) \rightarrow$
$UnSupp(b) \rightarrow eVal(\beta)\ UnSupp(a)\ UnSupp(\alpha)$

$UnSupp(b) \rightarrow eAcc(b)\ UnSupp(a)\ UnSupp(\alpha)$
$UnSupp(\alpha) \rightarrow UnSupp(b)$
$UnSupp(a) \rightarrow UnSupp(b)$
$eVal(\beta)\ eAcc(b) \rightarrow UnSupp(b)\ \}$
$\Sigma_d = \Sigma_{ss} \cup \{$
$Val(\alpha) \rightarrow UnSupp(b)\ UnSupp(\beta)\ \}$
$\Sigma_r = \Sigma_{ss} \cup \{$
$\rightarrow Acc(a)$
$\rightarrow Acc(b)$
$UnSupp(\beta) \rightarrow Val(\alpha)$
$UnSupp(b) \rightarrow Val(\alpha)$
$\rightarrow Val(\beta)\ \}$
$\Sigma_s = \Sigma_{ss} \cup \{$
$\rightarrow Acc(b)$
$\rightarrow Acc(a)$
$\rightarrow eVal(\beta)\ Val(\alpha)$
$\rightarrow eAcc(b)\ Val(\alpha)$
$\rightarrow Val(\beta)$
$\rightarrow Supp(a)\ UnSupp(a)$
$\rightarrow Supp(b)\ UnSupp(b)$
$\rightarrow Supp(\alpha)\ UnSupp(\alpha)$
$\rightarrow Supp(\beta)\ UnSupp(\beta)\ \}$

Here there exists a stable structure since α belongs to $UnSupp(U)$.
Prop 3 produces the following structures:

- *Conflict-free structures: there are 14 conflict-free structures among the 16 possible structures (are not conflict-free those that contain together b, α and β).*
- *Admissible structures:*

```
[]
[beta]
[a, beta]
[a]
```

- *Preferred structures:*

```
[a, beta]
```

- *Grounded structure:*

```
[a, beta]
```

- *Complete structures:*

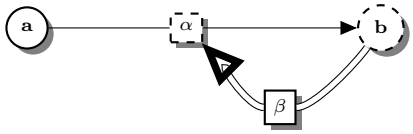
```
[a, beta]
```

- *Stable structures:*

```
[a, beta]
```

□

Example 29 *Consider the following REBAF.*



$\Sigma = \{$
$Supp(a)\ Acc(a) \rightarrow eAcc(a)$
$eAcc(a) \rightarrow Acc(a)$
$eAcc(a) \rightarrow Supp(a)$
$Supp(b)\ Acc(b) \rightarrow eAcc(b)$
$eAcc(b) \rightarrow Acc(b)$
$eAcc(b) \rightarrow Supp(b)$
$Supp(\alpha)\ Val(\alpha) \rightarrow eVal(\alpha)$
$eVal(\alpha) \rightarrow Val(\alpha)$
$eVal(\alpha) \rightarrow Supp(\alpha)$
$Supp(\beta)\ Val(\beta) \rightarrow eVal(\beta)$
$eVal(\beta) \rightarrow Val(\beta)$
$eVal(\beta) \rightarrow Supp(\beta)$
$eVal(\alpha)\ eAcc(a) \rightarrow NAcc(b)$
$NAcc(b)\ Acc(b) \rightarrow$
$\rightarrow Supp(a)$
$eVal(\beta)\ eAcc(b) \rightarrow Supp(\alpha)$
$\rightarrow Supp(\beta)\ \}$
$\Sigma_{ss} = \Sigma \cup \{$
$Supp(\alpha) \rightarrow eVal(\beta)$
$Supp(\alpha) \rightarrow eAcc(b)$
$Supp(b) \rightarrow$
$UnSupp(a) \rightarrow$
$\rightarrow UnSupp(b)$
$UnSupp(\beta) \rightarrow$
$UnSupp(\alpha) \rightarrow UnSupp(b)\ UnSupp(\beta)$
$UnSupp(\beta) \rightarrow UnSupp(\alpha)$
$UnSupp(b) \rightarrow UnSupp(\alpha)\ \}$
$\Sigma_d = \Sigma_{ss} \cup \{$
$Acc(b) \rightarrow UnSupp(a)\ UnSupp(\alpha)\ \}$
$\Sigma_r = \Sigma_{ss} \cup \{$
$\rightarrow Acc(a)$
$UnSupp(\alpha) \rightarrow Acc(b)$

$UnSupp(a) \rightarrow Acc(b)$
$\rightarrow Val(\alpha)$
$\rightarrow Val(\beta)$ }
$\Sigma_s = \Sigma_{ss} \cup \{$
$\rightarrow Acc(a)$
$\rightarrow eVal(\alpha)\ Acc(b)$
$\rightarrow eAcc(a)\ Acc(b)$
$\rightarrow Val(\alpha)$
$\rightarrow Val(\beta)$
$\rightarrow Supp(a)\ UnSupp(a)$
$\rightarrow Supp(b)\ UnSupp(b)$
$\rightarrow Supp(\alpha)\ UnSupp(\alpha)$
$\rightarrow Supp(\beta)\ UnSupp(\beta)$ }

Here there exists a stable structure since b belongs to $UnSupp(U)$.
Prop 3 produces the following structures:

- *Conflict-free structures: there are 14 conflict-free structures among the 16 possible structures (are not conflict-free those that contain together a, b and α).*
- *Admissible structures:*

```
[]
[beta]
[a, beta]
[a]
```

- *Preferred structures:*

```
[a, beta]
```

- *Grounded structure:*

```
[a, beta]
```

- *Complete structures:*

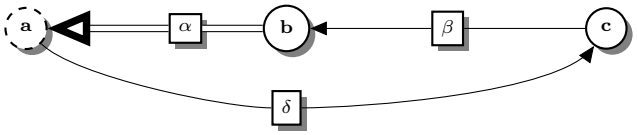
```
[a, beta]
```

- *Stable structures:*

```
[a, beta]
```

□

Example 30 *Consider the following REBAF.*



$\Sigma = \{$
$Supp(a)\ Acc(a) \rightarrow eAcc(a)$
$eAcc(a) \rightarrow Acc(a)$
$eAcc(a) \rightarrow Supp(a)$
$Supp(b)\ Acc(b) \rightarrow eAcc(b)$
$eAcc(b) \rightarrow Acc(b)$
$eAcc(b) \rightarrow Supp(b)$
$Supp(c)\ Acc(c) \rightarrow eAcc(c)$
$eAcc(c) \rightarrow Acc(c)$
$eAcc(c) \rightarrow Supp(c)$
$Supp(\alpha)\ Val(\alpha) \rightarrow eVal(\alpha)$
$eVal(\alpha) \rightarrow Val(\alpha)$
$eVal(\alpha) \rightarrow Supp(\alpha)$
$Supp(\beta)\ Val(\beta) \rightarrow eVal(\beta)$
$eVal(\beta) \rightarrow Val(\beta)$
$eVal(\beta) \rightarrow Supp(\beta)$
$Supp(\delta)\ Val(\delta) \rightarrow eVal(\delta)$
$eVal(\delta) \rightarrow Val(\delta)$
$eVal(\delta) \rightarrow Supp(\delta)$
$eVal(\delta)\ eAcc(a) \rightarrow NAcc(c)$
$NAcc(c)\ Acc(c) \rightarrow$
$eVal(\beta)\ eAcc(c) \rightarrow NAcc(b)$
$NAcc(b)\ Acc(b) \rightarrow$
$\rightarrow Supp(b)$
$\rightarrow Supp(c)$
$eVal(\alpha)\ eAcc(b) \rightarrow Supp(a)$
$\rightarrow Supp(\alpha)$
$\rightarrow Supp(\beta)$
$\rightarrow Supp(\delta)\ \}$
$\Sigma_{ss} = \Sigma \cup \{$
$Supp(a) \rightarrow eVal(\alpha)$
$Supp(a) \rightarrow eAcc(b)$
$UnSupp(b) \rightarrow$
$UnSupp(c) \rightarrow$
$UnSupp(a) \rightarrow eVal(\beta)\ UnSupp(b)\ UnSupp(\alpha)$
$UnSupp(a) \rightarrow eAcc(c)\ UnSupp(b)\ UnSupp(\alpha)$
$UnSupp(\alpha) \rightarrow UnSupp(a)$
$UnSupp(b) \rightarrow UnSupp(a)$
$eVal(\beta)\ eAcc(c) \rightarrow UnSupp(a)$
$UnSupp(\alpha) \rightarrow$
$UnSupp(\beta) \rightarrow$

$UnSupp(\delta) \rightarrow$ }

$\Sigma_d = \Sigma_{ss} \cup \{$
$Acc(c) \rightarrow UnSupp(a)\ UnSupp(\delta)$
$Acc(b) \rightarrow eVal(\delta)\ UnSupp(c)\ UnSupp(\beta)$
$Acc(b) \rightarrow eAcc(a)\ UnSupp(c)\ UnSupp(\beta)$ }

$\Sigma_r = \Sigma_{ss} \cup \{$
$\rightarrow Acc(a)$
$UnSupp(\beta) \rightarrow Acc(b)$
$UnSupp(c) \rightarrow Acc(b)$
$eVal(\delta)\ eAcc(a) \rightarrow Acc(b)$
$UnSupp(\delta) \rightarrow Acc(c)$
$UnSupp(a) \rightarrow Acc(c)$
$\rightarrow Val(\alpha)$
$\rightarrow Val(\beta)$
$\rightarrow Val(\delta)$ }

$\Sigma_s = \Sigma_{ss} \cup \{$
$\rightarrow Acc(a)$
$\rightarrow eVal(\beta)\ Acc(b)$
$\rightarrow eAcc(c)\ Acc(b)$
$\rightarrow eVal(\delta)\ Acc(c)$
$\rightarrow eAcc(a)\ Acc(c)$
$\rightarrow Val(\alpha)$
$\rightarrow Val(\beta)$
$\rightarrow Val(\delta)$
$\rightarrow Supp(a)\ UnSupp(a)$
$\rightarrow Supp(b)\ UnSupp(b)$
$\rightarrow Supp(c)\ UnSupp(c)$
$\rightarrow Supp(\alpha)\ UnSupp(\alpha)$
$\rightarrow Supp(\beta)\ UnSupp(\beta)$
$\rightarrow Supp(\delta)\ UnSupp(\delta)$ }

Prop 3 produces the following structures:

- *Conflict-free structures: there are 50 conflict-free structures among the 64 possible structures (are not conflict-free those that contain together a, c and δ, or b, c and β).*
- *Admissible structures:*

```
[ ]
[ alpha ]
[ alpha, beta ]
[ beta ]
[ alpha, beta, delta ]
[ delta ]
[ alpha, delta ]
[ beta, delta ]
```

```
[ c, beta, delta ]
[ c, beta ]
[ c, alpha, beta ]
[ c, alpha, beta, delta ]
[ a, b, alpha, beta, delta ]
[ a, b, alpha, delta ]
```

- *Preferred structures:*

```
[ c, alpha, beta, delta ]
[ a, b, alpha, beta, delta ]
```

- *Grounded structure:*

```
[ alpha, beta, delta ]
```

- *Complete structures:*

```
[ alpha, beta, delta ]
[ c, alpha, beta, delta ]
[ a, b, alpha, beta, delta ]
```

- *Stable structures:*

```
[ c, alpha, beta, delta ]
[ a, b, alpha, beta, delta ]
```

□