Logical encodings of interactions in an argumentation graph with recursive attacks

Claudette Cayrol, Marie-Christine Lagasquie-Schiex, Luis Fariñas del Cerro

Logical encodings of interactions in an argumentation graph with recursive attacks

Claudette Cayrol,
M-Christine Lagasquie-Schiex
Luis Farinas del Cerro

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

Tech. Report (Revised version)
IRIT/RR- -2017- -08- -FR

Abstract

This work is a preliminary study that proposes logical encodings for translating argumentation graphs *themselves* into logical knowledge bases. This translation will be used for identifying or redefining some properties of argumentation graphs. The graphs that we consider are used to formalize abstract argumentation with recursive attacks.

Contents

Chapter 1

Introduction

The main feature of argumentation framework is the ability to deal with incomplete and / or contradictory information, especially for reasoning [24, 3]. Moreover, argumentation can be used to formalize dialogues between several agents by modeling the exchange of arguments in, *e.g.*, negotiation between agents [4]. An argumentation system (AS) consists of a collection of arguments interacting with each other through a relation reflecting conflicts between them, called *attack*. The issue of argumentation is then to determine *acceptable* sets of arguments (*i.e.*, sets able to defend themselves collectively while avoiding internal attacks), called *extensions*, and thus to reach a coherent conclusion. Another form of analysis of an AS is the study of the particular status of each argument, this status is based on membership (or non-membership) in the extensions. Formal frameworks have greatly eased the modeling and study of AS. In particular, the framework of [24] allows completely abstracting the *concrete* meaning of the arguments and relies only on binary interactions that may exist between them.

AS have been extended along different lines. For instance, bipolar AS (BAS) correspond to AS with a second kind of interaction, the support relation. This relation represents a positive interaction between arguments and has been first introduced by [27, 38]. In [17], the support relation is left general so that the bipolar framework keeps a high level of abstraction. However there is no single interpretation of the support, and a number of researchers proposed specialized variants of the support relation (deductive support [10], necessary support [31, 32], evidential support [33, 34]). 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. In [18], a comparative study has been done in order to restate these proposals in a common setting, the bipolar argumentation framework. Basically, the idea is to keep the original arguments, to add complex attacks defined by the combination of the original attack and the support, and to modify the classical notions of acceptability. An important result of [18] is the highlight of a kind of duality between the deductive and the necessary interpretations of support, which results in a duality in the modelling by complex attacks. Recent works give a translation between necessary supports and evidential supports [36]; others propose a justification of the necessary support using the notion of subarguments [37]; an extension of the necessary support is presented in [30].

AS have been also extended so as to take into account interactions between arguments and other interactions. A first version has been introduced by [29], then studied in [6] under the name of AFRA

(Argumentation Framework with Recursive Attacks). This version describes abstract argumentation systems in which the interactions can be either attacks between arguments or attacks from an argument to another attack. In this case, as for the bipolar case, a translation of an AFRA into an equivalent AS can be defined by the addition of some new arguments and the attacks they produce or they receive. Recently, an extension of AFRA has been proposed in [21] in order to take into account supports on arguments or on interactions. These systems are called ASAF (Attack-Support Argumentation Frameworks). And, once again, a translation of an ASAF into an equivalent AS is proposed by the addition of arguments and attacks.

The subject of the current paper is to propose a logical vision of an ASAF that can justify the introduction of all these new attacks. This logical vision is initially issued from works in bioinformatics (see [23, 1]). In this domain, we can find *metabolic networks* that 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 [1]. This translation allows for the use of automated deduction methods for reasoning on these networks. A very preliminary work has been done for adapting this approach to argumentation systems (see [16]). Nevertheless, in terms of argumentation, the obtained results are limited at this time.

In this new work, we restrict our study to argumentation systems with only attacks (recursive or not) and propose a new logical vision of an AS and an ASAF.

Context. Two different cases are considered in this work:

- The case called "classic": we study argumentation systems, denoted by AS, where only attacks between arguments (*simple attacks*, see Definition 1 on page 5) can be found;
- The case called "recursive": we study argumentation systems, where there exist arguments, attacks between arguments and attacks between an argument and another attack, (called *recursive attacks*, see Definition 4 on page 6).

Note: In this work, the interactions corresponding to the notion of support will not not considered (this case is left for future work, see Section 7 on page 63).

According to the considered case (classic or recursive), there exist two ways for weakening an attack:

- either by weakening the source of the attack (this is possible in both cases),
- or because the attack is the target of another attack (this is possible only in the recursive case).

This leads to propose the notions of "grounded attack" and "valid attack" ([15]). The notion of grounded attack is about the source of the attack and the notion of valid attack is about the link between the source and the target of the attack (*i.e.* the role of the interaction itself).

Moreover, in [15], the argumentation graph is translated into a graph containing only simple attacks, using the addition of meta-arguments. This translation allows taking into account the notion of grounded (resp. valid) attack in the computation of the extensions of the resulting graph.

Contents:

1. The first two chapters contain the necessary background: the main notions about abstract argumentation are given in Chapter 2 on page 5 and some existing works about argumentation and logics are described in Chapter 3 on page 13.

2. In Chapter 4 on page 19, we propose a translation of the argumentation graph itself using a formal language: our aim is to formally express the semantics[1] attached to an attack in an argumentation graph, and so the notions of accepted argument, grounded or valid attack, independently of any argumentation semantics. The proposed formal language will first take into account simple attacks (Section 4.1 on page 19), then will be extended to cope with recursive attacks (Section 4.2 on page 23).

3. Then, using the formal language described in the previous chapter, a modelization of argumentation semantics is given in Chapter 5 on page 33. Our aim is to express the standard semantics (grounded, preferred, complete, stable) of an AS in logical terms (either by formulae, or by a selection of models). This will be done with the basic language (Section 5.1 on page 33) and with the extended language (Section 5.2 on page 44).

4. This will suggest the definition of new argumentation semantics capable of handling recursive attacks (Chapter 6 on page 51).

Notes:

- A similar work has been done in [16]. However, the proposed encoding presented here is not the same as the one given in [16] and so the results of [16] are not usable here.
- All the examples used in this report are totally described in Appendix A on page 67.
- Proofs are given in Appendix B on page 115.

[1] In the common sense of this word and not in the sense of "argumentation semantics" evoked in Section 2.2 on page 7.

Chapter 2

Background on abstract argumentation

This chapter gives some definitions about abstract argumentation (the different kinds of argumentation systems and the standard argumentation semantics).

2.1 Different abstract argumentation systems

The classic case concerns argumentation systems with only one kind of interaction: attacks between arguments.

Def. 1 (AS) *An* argumentation system (AS) *is a tuple* $\langle \mathbf{A}, \mathbf{R} \rangle$, *where*

- $\mathbf{A}$ *is a finite and non-empty set of arguments,*
- $\mathbf{R} \subseteq \mathbf{A} \times \mathbf{A}$ *is an* attack relation.

A first extension takes into account an additional kind of interaction: supports between arguments.

Def. 2 (BAS) *A* bipolar argumentation system (BAS) *is a tuple* $\langle \mathbf{A}, \mathbf{R}_{\text{att}}, \mathbf{R}_{\text{sup}} \rangle$, *where*

- $\mathbf{A}$ *is a finite and non-empty set of arguments,*
- $\mathbf{R}_{\text{att}} \subseteq \mathbf{A} \times \mathbf{A}$ *is an* attack relation *and*
- $\mathbf{R}_{\text{sup}} \subseteq \mathbf{A} \times \mathbf{A}$ *is a* support relation.

It is assumed that $\mathbf{R}_{\text{att}} \cap \mathbf{R}_{\text{sup}} = \varnothing$.

Another possible extension concerns recursive interactions (support or attack), *i.e.* from an argument to either another argument or another interaction [22].

Def. 3 (ASAF) *An* Attack-Support Argumentation Framework (ASAF) *is a tuple* $\langle \mathbf{A}, \mathbf{R}_{\text{att}}, \mathbf{R}_{\text{sup}} \rangle$ *where*

- $\mathbf{A}$ *is a set of arguments,*

- $\mathbf{R}_{\text{att}}$ *is a subset of* $\mathbf{A} \times (\mathbf{A} \cup \mathbf{R}_{\text{att}} \cup \mathbf{R}_{\text{sup}})$ *corresponding to a set of attacks, and*
- $\mathbf{R}_{\text{sup}}$ *is a subset of* $\mathbf{A} \times (\mathbf{A} \cup \mathbf{R}_{\text{att}} \cup \mathbf{R}_{\text{sup}})$ *corresponding to a set of supports. Note that* $\mathbf{R}_{\text{sup}}$ *is assumed to be irreflexive and transitive.*

It is assumed that $\mathbf{R}_{\text{att}} \cap \mathbf{R}_{\text{sup}} = \varnothing$.

We propose an alternative formalisation in which each interaction is labelled.[1]

Def. 4 (ASAF) *An ASAF is a tuple* $\langle \mathbf{A}, \mathbf{R}_{\text{att}}, \mathbf{R}_{\text{sup}}, s, t \rangle$ *where*

- $\mathbf{A}$ *is a set of arguments,*
- $\mathbf{R}_{\text{att}}$ *(resp.* $\mathbf{R}_{\text{sup}}$*) is a set disjunct from* $\mathbf{A}$*, representing attacks (resp. supports),*
- s *is a function from* $(\mathbf{R}_{\text{att}} \cup \mathbf{R}_{\text{sup}})$ *to* $\mathbf{A}$*, mapping each interaction to its source,*
- t *is a function from* $(\mathbf{R}_{\text{att}} \cup \mathbf{R}_{\text{sup}})$ *to* $(\mathbf{A} \cup \mathbf{R}_{\text{att}} \cup \mathbf{R}_{\text{sup}})$ *mapping each interaction to its target.*

It is assumed that $\mathbf{R}_{\text{att}} \cap \mathbf{R}_{\text{sup}} = \varnothing$.

Note that each interaction α can be identified with the pair $(s(\alpha), t(\alpha))$.

Note also that a BAS can be recovered as a particular case where t is a mapping from $(\mathbf{R}_{\text{att}} \cup \mathbf{R}_{\text{sup}})$ to $\mathbf{A}$.

In the remainder of this paper, we do not consider the support relation. So we consider either an AS, or an ASAF with $\mathbf{R}_{\text{sup}} = \varnothing$. In both cases, the attack relation will be denoted by $\mathbf{R}$.

So, we consider a 4-tuple $\langle \mathbf{A}, \mathbf{R}, s, t \rangle$ where

- $\mathbf{A}$ is a set of arguments (denoted by $a, b, \ldots$) and $\mathbf{R}$ is a set disjunct from $\mathbf{A}$, representing attacks (denoted by greek letters),
- s is a function from $\mathbf{R}$ to $\mathbf{A}$, mapping each interaction to its source,
- t is a function from $\mathbf{R}$ to $(\mathbf{A} \cup \mathbf{R})$ mapping each interaction to its target.

An AS can be recovered as a particular case where t is a mapping from $\mathbf{R}$ to $\mathbf{A}$.

For each of these argumentation systems, we use its graphical representation defined by $G = (\mathbf{A}, \mathbf{R})$ ($\mathbf{A}$ being the set of vertices and $\mathbf{R}$ being the set of edges).

As usually done in Graph Theory, we adopt the following notations: let $a \in \mathbf{A}$, $\mathbf{R}^+(a)$ (resp. $\mathbf{R}^-(a)$) denotes the set of arguments attacked by (resp. that attack) a.

[1] Another formalisation is also given in [15]: A *labelled ASAF (LASAF)* is a 5-uple $\langle \mathbf{A}, \mathbf{R}_{\text{att}}, \mathbf{R}_{\text{sup}}, \mathcal{V}, \mathcal{L} \rangle$ where $\mathbf{A}$ is a set of arguments, $\mathbf{R}_{\text{att}} \subseteq \mathbf{A} \times (\mathbf{A} \cup \mathbf{R}_{\text{att}} \cup \mathbf{R}_{\text{sup}})$ is an attack relation, $\mathbf{R}_{\text{sup}} \subseteq \mathbf{A} \times (\mathbf{A} \cup \mathbf{R}_{\text{att}} \cup \mathbf{R}_{\text{sup}})$ is a support relation, $\mathcal{V}$ is a set of labels (denoted by greek letters) and $\mathcal{L}$ is a bijection from $\mathbf{R} \subseteq (\mathbf{R}_{\text{att}} \cup \mathbf{R}_{\text{sup}})$ to $\mathcal{V}$.

2.2 Argumentation semantics for AS

In the following, we recalled the definitions of the standard semantics. Note that these definitions only concern classic abstract argumentation systems (with simple attacks). It is important to note that no semantics has been clearly defined in the other cases, *i.e.* when supports and/or recursive interactions exist (some suggestions have been done but there exists no consensus about them).

In the extension-based approach, a semantics specifies requirements that a set of arguments should satisfy. These requirements have been extensively analysed in [BG07]. There are three basic requirements, corresponding to three principles for semantics:

- An extension is a set of arguments that "can stand together". This corresponds to the *conflict-free principle*.
- An extension is a set of arguments that "can stand on its own", namely is able to counter all the attacks it receives. This corresponds to the *concept of defence* and leads to the *admissibility principle*.
- *Reinstatement* is a kind of dual principle. If an extension defends an argument, this argument is reinstated by the extension and should belong to the extension.

We give below the formal definitions.

Def. 5 (Basic concepts used in extension-based semantics) *Let AS* = $\langle \mathbf{A}, \mathbf{R} \rangle$ *and* $S \subseteq \mathbf{A}$.

- S *is* conflict-free *if and only if there are no arguments* $a, b \in S$*, such that* $a\mathbf{R}b$.
- $a \in \mathbf{A}$ *is* acceptable *with respect to* S *(or equivalently* S *defends* a*) if and only if* $\forall b \in \mathbf{A}$ *such that* $b\mathbf{R}a$*,* $\exists c \in S$ *such that* $c\mathbf{R}b$.
- S *is* admissible *if and only if* S *is conflict-free and each argument in* S *is acceptable with respect to* S.

Standard extension-based semantics are classically defined as follows:

Def. 6 (AS extensions in standard semantics) *Let AS* = $\langle \mathbf{A}, \mathbf{R} \rangle$ *and* $S \subseteq A$.

- S *is* a naive extension *of AS if and only if it is a maximal (with respect to* $\subseteq$*) conflict-free set.*
- S *is* a preferred extension *of AS if and only if it is a maximal (with respect to* $\subseteq$*) admissible set.*
- S *is* a complete extension *of AS if and only if* S *is admissible and each argument which is acceptable with respect to* S *belongs to* S.
- S *is* a stable extension *of AS if and only if it is conflict-free and* $\forall a \notin S$*,* $\exists b \in S$ *such that* $b\mathbf{R}a$.

Most of the standard semantics can be alternatively defined using the characteristic function $\mathcal{F}$.

Def. 7 (Extensions defined by $\mathcal{F}$) *Let AS* = $\langle \mathbf{A}, \mathbf{R} \rangle$ *and* $S \subseteq A$.

- *The* characteristic function *of AS is defined by:* $\mathcal{F}(S) = \{a \in \mathbf{A}$ *such that* a *is acceptable with respect to* $S\}$.
- S *is* admissible *if and only if* S *is conflict-free and* $S \subseteq \mathcal{F}(S)$.
- S *is* a complete extension *of AS if and only if it is conflict-free and a fixed point of* $\mathcal{F}$.
- S *is* the grounded extension *of AS if and only if it is the minimal (with respect to* $\subseteq$*) fixed point*[2] *of* $\mathcal{F}$.
- S *is* a preferred extension *of AS if and only if it is a maximal (with respect to* $\subseteq$*) complete extension if and only if it is a maximal conflict-free fixed point of* $\mathcal{F}$.

Note that due to Definition 6 on the previous page the complete semantics is based on both principles of admissibility and resinstatement. Moreover, as the grounded extension, the preferred extensions and the stable extensions are also complete extensions, the grounded (resp. preferred, stable) semantics satisfies the admissibility and reinstatement principles.

2.3 Argumentation semantics for BAS

Handling support and attack at an abstract level has the advantage to keep genericity and to give an analytic tool for studying complex attacks and new semantics considering both attack and support relations, among others. However, the drawback is the lack of guidelines for choosing the appropriate definitions and semantics depending on the application. For solving this problem, some variants of the support relation have been proposed recently, including the necessary support. This kind of support was initially proposed in [32] with the following interpretation: If $c\mathbf{R}_{\text{sup}}b$ then the acceptance of c is necessary to get the acceptance of b, or equivalently the acceptance of b implies the acceptance of c. Suppose now that $a\mathbf{R}_{\text{att}}c$. The acceptance of a implies the non-acceptance of c and so the non-acceptance of b. Also, if $c\mathbf{R}_{\text{sup}}a$ and $c\mathbf{R}_{\text{att}}b$, the acceptance of a implies the acceptance of c and the acceptance of c implies the non-acceptance of b. So, the acceptance of a implies the non-acceptance of b. These constraints relating a and b are enforced by adding new complex attacks from a to b:

Def. 8 ([32] Extended attack) *Let* $\langle \mathbf{A}, \mathbf{R}_{\text{att}}, \mathbf{R}_{\text{sup}} \rangle$ *and* $a, b \in \mathbf{A}$. *There is an* extended attack *from* a *to* b *iff*

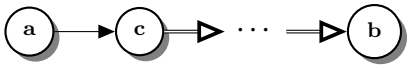
- *either (1)* $a\mathbf{R}_{\text{att}}b$*;*
- *or (2)* $a_1\mathbf{R}_{\text{att}}a_2\mathbf{R}_{\text{sup}} \ldots \mathbf{R}_{\text{sup}}a_n$, $n \geq 3$, *with* $a_1 = a$, $a_n = b$*;*
- *or (3)* $a_1\mathbf{R}_{\text{sup}} \ldots \mathbf{R}_{\text{sup}}a_n$, *and* $a_1\mathbf{R}_{\text{att}}a_p$, $n \geq 2$, *with* $a_n = a$, $a_p = b$.

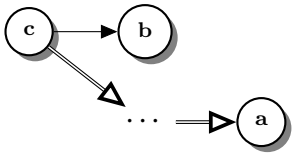
Consider the following graphical notations:

the simple edges	$\longrightarrow$	represent attacks,
the double edges	$\Longrightarrow$	represent supports.

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

The following figures illustrate the cases 2 and 3 of Definition 8 on the facing page:

Case 2: 

Case 3: 

Among the frameworks proposed in [19] for handling necessary supports, we focus on the one encoding the following interpretation: If $c\mathbf{R}_{\text{sup}}b$, "the acceptance of c is necessary to get the acceptance of b" because "c is the *only* attacker of a particular attacker of b":

Def. 9 ([19] MAS associated with a BAS) *Let $\langle \mathbf{A}, \mathbf{R}_{\text{att}}, \mathbf{R}_{\text{sup}} \rangle$ be a BAS with $\mathbf{R}_{\text{sup}}$ being a set of necessary supports. Let $\mathbf{A}_n = \{N_{cb} | (c,b) \in \mathbf{R}_{\text{sup}}\}$ and $\mathbf{R}_n = \{(c, N_{cb}) | (c,b) \in \mathbf{R}_{\text{sup}}\} \cup \{(N_{cb}, b) | (c,b) \in \mathbf{R}_{\text{sup}}\}$. The tuple $\langle \mathbf{A} \cup \mathbf{A}_n, \mathbf{R}_{\text{att}} \cup \mathbf{R}_n \rangle$ is the meta-argumentation system (MAS) associated with the BAS (it is a Dung's AS).*

Since a MAS is a Dung's AS, any Dung's semantics can be applied.

2.4 Argumentation semantics for ASAF

Here, as the bipolar case, there is no consensus about semantics for ASAF. At least four distinct methods exist. The first third ones consist in a translation of the original ASAF into an AS (in which all Dung's semantics can be reused) whereas the last one gives direct definitions for ASAF semantics without using a translation into an AS.

2.4.1 Method of [6]

The proposed translation uses the notion of *defeat* defined as follows:[3]

Def. 10 ([6] Defeat in ASAF without support) *Let $\langle \mathbf{A}, \mathbf{R} \rangle$ be an ASAF without support. Let $\alpha, \beta \in \mathbf{R}$. Let $X \in \mathbf{A} \cup \mathbf{R}$.*

- α directly defeats X *iff* X *is the target of* α.
- α indirectly defeats β *iff the target of* α *is an argument that is the source of* β.

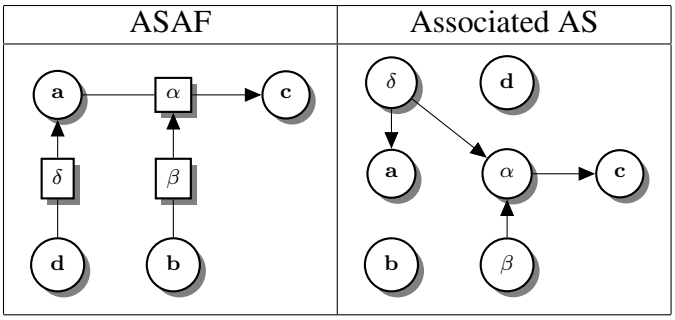
Then, a translation of an ASAF into an AS is provided:

Def. 11 ([6] AS associated with an ASAF) *Let $\langle \mathbf{A}, \mathbf{R} \rangle$ be an ASAF without support. The AS associated with this ASAF is $\langle \mathbf{A}', \mathbf{R}' \rangle$ defined by:*

- $\mathbf{A}' = \mathbf{A} \cup \mathbf{R}$,
- $\mathbf{R}' = \{(X, Y)$ *s.t.* $X \in \mathbf{R}, Y \in \mathbf{A} \cup \mathbf{R}$ *and* X *directly or indirectly defeats* $Y\}$.

The previous notions are illustrated on the following example:

[3] Initially in [6], the used argumentation system with recursive attacks is called an AFRA.



For instance, α directly defeats c, β directly defeats α and δ indirectly defeats α.

The following points seem counterintuitive:

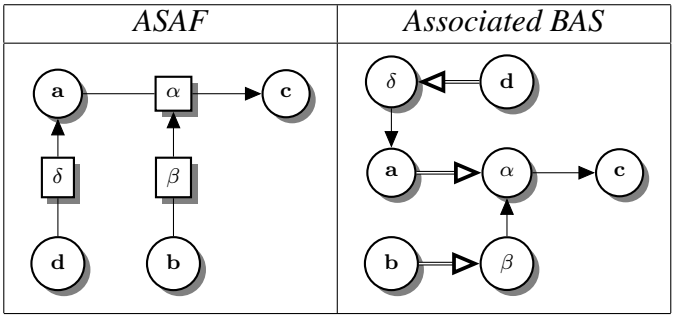
- there is no attack between a and c (more generally, no argument from $\mathbf{A}$ can be an attacker in the associated AS of the ASAF),
- there is no link between a and α (more generally, there is no link between an attack and its source); that is surprizing since, without a, the attack α does not exist.

2.4.2 Method of [21]

The translation from an ASAF into an AS follows two steps:[4]

1. First, the ASAF is turned into a BAS with necessary support.
2. Then, this BAS is turned into an AS by adding extended attacks.

Def. 12 ([21] BAS associated with ASAF) *The following schemas describe the encoding of attacks (attacked or not):*

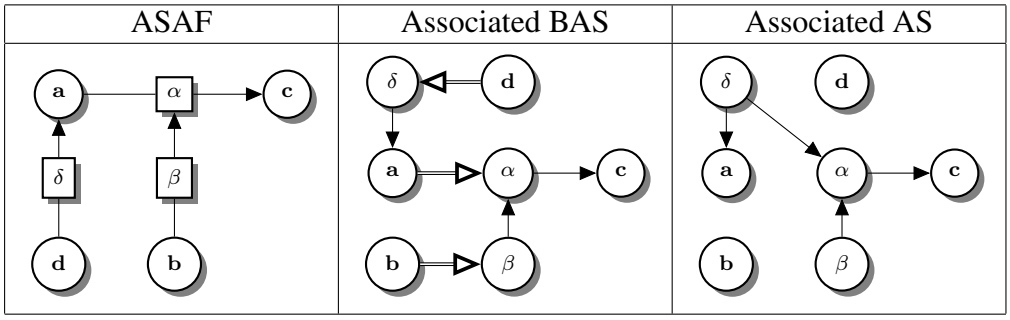


Given the BAS associated with the ASAF, the second step followed in [21] is to create an AS by adding complex attacks, namely Case 2 - extended attacks (see Def. 8):

Def. 13 ([21] AS associated with BAS and ASAF) *Let $\langle \mathbf{A}, \mathbf{R}_{\text{att}}, \mathbf{R}_{\text{sup}} \rangle$ be the BAS associated with a given ASAF. The pair $\langle \mathbf{A}', \mathbf{R}' \rangle$, where $\mathbf{A}' = \mathbf{A}$ and $\mathbf{R}' = \mathbf{R}_{\text{att}} \cup \{(a, b) |$ there is a sequence $a_1\ \mathbf{R}_{\text{att}}\ a_2\ \mathbf{R}_{\text{sup}} \ldots \mathbf{R}_{\text{sup}}\ a_n, n \geq 3,$ with $a_1 = a, a_n = b\}$ is the AS associated with the BAS and the ASAF.*

[4] Initially in [21], the used argumentation system contains recursive attacks and supports and the definitions are more complex in order to take into account supports.

For instance (in this case the attack from δ to α is added following Definition 13 on the preceding page):



2.4.3 Method of [15]

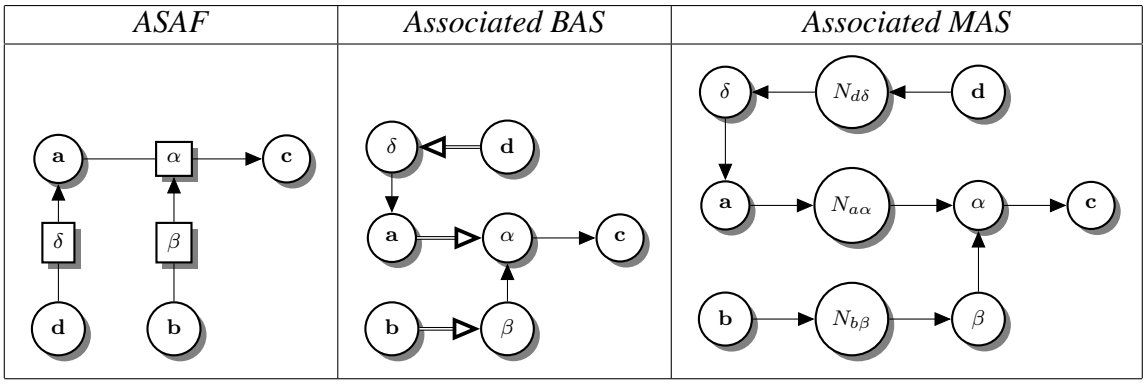
The proposed translation consists in encoding recursive interactions into a MAS in two steps:[5]

1. First, the ASAF is turned into a BAS with necessary support (exactly in the same way that is done in [21]).

2. Then, this BAS is turned into an MAS following Definition 9 on page 9.

We give here only the definition corresponding to an ASAF without support:

Def. 14 ([15] Encoding of labelled attacks) *The following schemas describe the encoding of a labelled attack (attacked or not):*



Note that the N_{ij} code the ground-links, *i.e.* the links between the source of an interaction and the interaction.

[5]Initially in [15], the used argumentation system is called a LASAF and contains recursive attacks and supports; so the definitions are more complex in order to take into account supports.

2.4.4 Method of [22]

Here the first idea is similar to the one developed in [6]: introducing a notion of *defeat*. Here we only give the definitions corresponding to an ASAF without support, *i.e.* the definitions relied to the notion of *unconditional defeat*.[6]

The unconditional defeats can be partitioned into two distinct cases: the first case corresponds to conflicts already captured by the attack relation of the ASAF (called direct defeats), the second case (called indirect defeat) captures the intuition that attacks are strictly related to their source, as in [6].

Def. 15 ([22] Defeats) *Let* $\langle \mathbf{A}, \mathbf{R} \rangle$ *be an ASAF without support,* $\alpha, \beta \in \mathbf{R}$ *and* $X \in \mathbf{A} \cup \mathbf{R}$.

Direct defeat α *directly defeats* X*, noted* α `d-def` X*, iff the target of* α *is* X.

Indirect defeat α *indirectly defeats* β*, noted* α `i-def` β*, iff* α `d-def` *the source of* β.

Unconditional defeat α *unconditionally defeats* X*, noted* α `u-def` X*, iff* α `d-def` X *or* α `i-def` X.

Then a redefinition of the main concepts used in argumentation semantics is given:[7]

Def. 16 ([22] Main semantics concepts for semantics) *Let* $\langle \mathbf{A}, \mathbf{R} \rangle$ *be an ASAF without support,* $S \in \mathbf{A} \cup \mathbf{R}$.

Conflict-Freeness S *is conflict-free iff:* $\nexists \alpha, X \in S$ *s.t.* α `u-def` X.

Acceptability *Let* $X \in \mathbf{A} \cup \mathbf{R}$*,* X *is acceptable wrt* S *iff:* $\forall \alpha \in \mathbf{R}$ *s.t.* α `u-def` X*,* $\exists \beta \in S$ *s.t.* β `u-def` α.

Then, using these new notions of conflict-freeness and acceptability as in Dung's definitions, it is possible to redefine all classical semantics.

Even if this approach is methodologically distinct from the one of [6], in the case of an ASAF without support, the same results are obtained.

[6] Initially in [22], the used argumentation system contains recursive attacks and supports and the definitions are more complex in order to take into account supports.

[7] Once again we only give the part of definition corresponding to an ASAF without support.

Chapter 3

Argumentation and logics

Correspondences between argumentation and logic programming have been introduced in the fundamental article of Dung [24]. From that moment, several works have been done (see for instance [14, 35, 13] and more references can be found in this last article).

The most studied problem is the following: find an appropriate encoding of an argumentation system AS into a logic program P and its associated logic programming semantics, such that, applied to P, these semantics capture argumentation semantics of the original AS.

For that purpose, many different encodings have been proposed. For instance Dung [24] has proposed an encoding allowing the capture of (only) grounded and stable semantics. In [14, 35, 13], the proposed encodings allow for the characterization the standard argumentation semantics (grounded, stable, preferred and complete semantics). The main difference between [14, 35] and [13] is that, in [14, 35], logic programming semantics are defined using classical models (2-valued semantics), whereas, in [13], 3-valued semantics are used.

The following section recalls the main basic concepts of logic programming, with the definition of 2-valued and 3-valued semantics. Then we focus on [14, 35, 13] and present the main characteristics of each encoding.

3.1 Background on logic programming

Let L be a finite set of symbols (or atoms). A literal is an atom a (positive literal) or the negation of an atom $not\ a$ (negative literal).

If $\{a_1, \ldots, a_n\}$ is a set of atoms, $not\{a_1, \ldots, a_n\}$ denotes the set of negative literals $\{not\ a_1, \ldots, not\ a_n\}$.

A normal rule r is an expression of the form:

$$c \leftarrow a_1, \ldots, a_n, not\ b_1, \ldots, not\ b_m$$

where c, a_i and b_j are atoms.

Notations: $B^+(r) = \{a_1, \ldots, a_n\}$ and $B^-(r) = \{not\ b_1, \ldots, not\ b_m\}$.

A normal logic program is a finite set of normal rules. L_P denotes the set of the atoms appearing in P.

Note about the negation
According to the way the logic program is taken into account, not represents either the "failure negation" (as in [13]) or the classical negation (as in [14, 35]). In this last case, a logic program can be considered as a classical logic theory and the literal $not\ a$ can be replaced by $\neg a$.

2-valued semantics in logic programming
Let P be a normal logic program (considered as a classical theory) and M be a set of atoms, $M \subseteq L_P$. The p-stable models and the stable models are defined with reduction operations.

Def. 17 *The reduction of P by M is the program $Red(P, M)$ defined as follows: If r is a rule of P under the form $c \leftarrow a_1, \ldots, a_n, \neg b_1, \ldots, \neg b_m$, r is modified by removing all literals $\neg b_i$ such that $b_i \notin M$.*
M is a p-stable model of P if and only if M is a model of $Red(P, M)$ and $Red(P, M) \vdash a$ for all $a \in M$.

Another kind of reduction is used for defining the stable models.

Def. 18 *P^M is the program obtained from P by the removal of:*

- *any rule r under the form $c \leftarrow a_1, \ldots, a_n, \neg b_1, \ldots, \neg b_m$ with $b_i \in M$, then*
- *any negative literal into the remaining rules.*

M is a stable model of P if and only if M is a minimal model of P^M.

Semantics of "supported" models are defined as follows:

Def. 19 *M is a supported model of P if and only if, for all $a \in M$, there exists a rule r under the form $a \leftarrow a_1, \ldots, a_n, \neg b_1, \ldots, \neg b_m$ such that $\{a_1, \ldots, a_n\} \subseteq M$ and $\{b_1, \ldots, b_m\} \cap M = \varnothing$.*

The supported models of a program P are exactly the models obtained by the Clack completion of P.

Def. 20 *The Clark completion of the program P is the program $Comp(P)$ obtained following the two next steps:*

- *any rule r under the form $c \leftarrow a_1, \ldots, a_n, \neg b_1, \ldots, \neg b_m$ is replaced by the formula $c \leftarrow a_1 \wedge \ldots \wedge a_n \wedge \neg b_1 \wedge \ldots \wedge \neg b_m$.*
- *$\forall a \in L_P$, $Support(a)$ denotes the set of formulae whose conclusion is a (so $Support(a) = \{a \leftarrow \phi_1, \ldots, a \leftarrow \phi_k\}$). $Support(a)$ is replaced by the unique formula $a \leftrightarrow (\phi_1 \vee \ldots \vee \phi_k)$. If $Support(a)$ is empty, we add the formula $\neg a$ to $Comp(P)$.*

Example:
$P = \{p \leftarrow a, p \leftarrow b, a \leftarrow \neg b, b \leftarrow \neg a\}$. $Comp(P) = \{p \leftrightarrow a \vee b, a \leftrightarrow \neg b, b \leftrightarrow \neg a\}$. There are two models: $\{a, p\}$ and $\{b, p\}$.

3-valued semantics of logic programming
Let P be a normal logic program. A 3-valued assignment I of P is a pair (T, F) of sets of atoms ($T \subseteq L_P$, $F \subseteq L_P$) such that $T \cap F = \varnothing$. Intuitively, atoms of T are *true* in I, those of F are *false* in I and those of $L_P \setminus (T \cap F)$ are undefined ("undef").
The reduction of P by I is defined as the program denoted by P/I and obtained as follows:

- any rule r under the form $c \leftarrow a_1, \ldots, a_n, not\ b_1, \ldots, not\ b_m$ with $b_i \in T$ is removed.
- for each remaining rule, we modify the rule by removing the literals $not\ b_i$ such that $b_i \in F$.
- for each remaining rule, any literal $not\ b_i$ is replaced by the symbol u (u is an atom $\notin L_P$ assumed undefined in any assignment).

The reduced program P/I has a unique 3-valued model (T, F) with T minimal and F maximal (obtained by a fixed-point operator, see also [13] for an equivalent definition of the smallest model), denoted by $\Gamma_P(I)$.

The 3-valued semantics are defined using Γ_P.

- I is a p-stable 3-valued model of P iff $\Gamma_P(I) = I$.
- $I = (T, F)$ is a well-founded 3-valued model of P iff I is a p-stable 3-valued model of P with T minimal.
- $I = (T, F)$ is a regular 3-valued model of P iff I is a p-stable 3-valued model of P with T maximal.
- $I = (T, F)$ is a stable 3-valued model of P iff I is a p-stable 3-valued model of P with $T \cup F = L_P$.

3.2 Encoding of [14, 35]

$\Pi(G)$ will denote the logic program associated with the argumentation graph G.

Let $G = (\mathbf{A}, \mathbf{R})$. $\Pi(G)$ is defined by associating:

- any argument x with a literal $def(x)$ ("x is defeated") whose meaning is "x cannot belong to an admissible set" and with a literal $acc(x)$ ("x is acceptable") whose meaning is "x can be considered as accepted"
- any argument x with the set denoted by $\Pi(x)$ containing the following normal rules:
  - for any $b\mathbf{R}x$, the rule $def(x) \leftarrow \neg def(b)$
  - for any $b\mathbf{R}x$, the rule $def(x) \leftarrow def(c_1) \wedge \ldots \wedge def(c_k)$ with $\{c_1, \ldots, c_k\} = \mathbf{R}^-(b)$

The first part of $\Pi(x)$ encodes the fact that an argument is defeated if at least one of its attackers is not defeated. The second part of $\Pi(x)$ encodes the fact that an argument is defeated if all its defenders against an attacker are defeated.

The normal program $\Pi(G)$ associated with the graph G is obtained by the union of $\Pi(x)$, for $x \in A$.

Notes: The first part of the program $\Pi(G)$ only encodes the notion of defeated arguments. The second part encodes the notion of acceptability.

Then, the encoding is completed in order to identify the acceptable arguments. The idea is the following: if the atom $def(x)$ is *false* in a classical model of $\Pi(G)$, then x can be considered as accepted. This idea is translated into the following rule: $acc(x) \leftarrow \neg def(x)$.
In the remainder of this paper, the obtained program will be denoted by $\Pi'(G)$.

Notes: [35] proves that any theorem of $\Pi(G)$ is a theorem of $\Pi'(G)$ and that any theorem of $\Pi'(G)$ belonging to the language of $\Pi(G)$ is already a theorem of $\Pi(G)$.

Example: $\mathbf{A} = \{a, b, c\}$, $\mathbf{R} = \{(a, b), (b, c)\}$. $\Pi'(G)$ contains:
$def(b) \leftarrow \neg def(a)$
$def(c) \leftarrow \neg def(b)$
$def(b) \leftarrow \top$ (the conjonction of an empty set corresponds to the tautology denoted by $\top$)
$def(c) \leftarrow def(a)$
$acc(a) \leftarrow \neg def(a)$, $acc(b) \leftarrow \neg def(b)$, $acc(c) \leftarrow \neg def(c)$.

Characterization of argumentation semantics

Let $E \subseteq \mathbf{A}$ be a subset of arguments. $Acc(E)$ denotes the set of literals $acc(x)$ for $x \in E$. We define $m(E) = Acc(E) \cup \{def(x), x \notin E\}$.

The results of [14] are completed by [35] in order to characterize the stable extensions (resp. preferred, complete) of an AS represented by a graph G, using models of the program $\Pi'(G)$ under different semantics of logic programming.

- E is a stable extension of G iff $m(E)$ is a stable model of $\Pi'(G)$
- E is a preferred extension of G iff $m(E)$ is a p-stable model of $\Pi'(G)$
- E is a complete extension of G iff $m(E)$ is a supported model of $\Pi'(G)$ (iff $m(E)$ is a model of the Clark completion of $\Pi'(G)$)

There also exists a direct characterization of the grounded extension but using models in a 3-valued logic (well-founded semantics).

3.3 Encoding of [13]

Let $G = (\mathbf{A}, \mathbf{R})$. For defining $\Pi(G)$, each argument x is associated with the rule $r(x)$ under the form $x \leftarrow not\ b_1, \ldots, not\ b_m$ with $\{b_1, \ldots, b_m\} = \mathbf{R}^-(x)$. If x is unattacked, $r(x)$ is under the form $x \leftarrow \top$.

The normal program $\Pi(G)$ associated with the graph G is obtained by the union of $\Pi(x)$, for $x \in A$.

Example: $\mathbf{A} = \{a, b, c\}$, $\mathbf{R} = \{(b, a), (c, a)\}$.
$\Pi(G)$ contains the 3 following rules: $b \leftarrow \top$, $c \leftarrow \top$, $a \leftarrow not\ b, not\ c$.

There exists a bijection between the 3-valued assignments of $\Pi(G)$ and the labellings of G. A labelling of G is an application from $\mathbf{A}$ to $\{in, out, undec\}$. Most of argumentation semantics can be expressed using labellings. So a characterization of argumentation semantics can be obtained in term of 3-valued models of the program $\Pi(G)$.

Characterization of argumentation semantics

- Complete extensions of G are obtained using p-stable 3-valued models of $\Pi(G)$
- Preferred extensions of G are obtained using regular 3-valued models of $\Pi(G)$
- The grounded extension G is obtained using the well-founded 3-valued model of $\Pi(G)$
- Stable extensions of G are obtained using stable 3-valued models of $\Pi(G)$

Notes

- The encoding proposed by [13] is very simple and produces a logic program with a weak negation.
- So acceptability is encoded by default.
- However, semantics are defined through 3-valued models and not with classical logic.

3.4 Other related works

The issue of logical encoding of argumentation has recently been addressed for different purposes independently of the notion of logic programming.

[25] propose a first-order logical language for expressing dynamics of an AS. A distinguished predicate symbol enables to code an attack between sets of arguments so that an attack is encoded by an atomic formula of the language.

[2] propose a formal language built upon a classical propositional language for representing various forms of structured arguments. Several kinds of interaction between these arguments can be captured as inference rules in the language. This approach allows for the representation of nested interactions but does not account for acceptability issues.

Many other works (**the majority in this domain**) concern the computation of extensions: [9] encode semantics of attack graphs by logical formulas (given a semantics σ and a set S of arguments, a formula is provided which is satisfiable iff S is a σ-extension); [26] does the same thing but with a modal logic considering that the accessibility relation is the inverse of the attack relation; the same kind of work is presented in [5] using signed theories and QBF formulae; [39] present algorithms using particular logical notions (minimal correction sets, backbone) in order to compute some semantics (semi-stable and eager); [20] translate complete labellings into logical formulae in order to compute preferred extensions with SAT solvers; [11] propose 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 [12], 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.

Moreover in the context of the First International Competition on Computational Models of Argumentation (ICCMA), different solvers have been proposed and tested (see for instance [7], or [28]). However, in all these works, neither the attack relation itself is logically encoded, nor the bipolar or the recursive aspects are taken into account.

Chapter 4

Graph description in a formal language

4.1 Preliminary version of the language for the classic case

Our aim is to describe an argumentation graph G containing only simple attacks. For this purpose, we consider a logical language in which we are able to represent arguments, and their properties (accepted or not, attacked or not, ...). $\Sigma^0(G)$ will denote the set of logical formulae describing the argumentation graph G.

4.1.1 Vocabulary

We use first-order logic (with the classical connectors $\vee$, $\wedge$, $\rightarrow$, $\leftrightarrow$, $\neg$, and the quantifiers $\exists$ and $\forall$). For defining $\Sigma^0(G)$, each argument x is associated with:

- a literal $Acc(x)$ ("x is accepted") and
- a literal $NAcc(x)$ ("x is attacked by an accepted argument").

Notes: The meaning of $NAcc(x)$ is not "x is not accepted" (so $NAcc(x)$ is not logically equivalent to $\neg Acc(x)$), but rather "x cannot be accepted" since "x is attacked by an accepted argument". Then taking into account the semantics of an attack leads to deduce $\neg Acc(x)$ from $NAcc(x)$ (see Section 4.1.2).

4.1.2 Properties

The property given below describes the behaviour of a simple attack (from an argument to another one) in an argumentation graph, *i.e.* the impact of an existing attack on the arguments that it involves. This property, denoted by $\mathbf{P}^0_{\mathbf{acc}}$, translates the semantics[1] of an attack edge between two arguments into a constraint on the acceptability of the arguments involved in the attack.

[1] Always in the first sense of this word and not in the sense of the argumentation semantics evoked in Section 2.2 on page 7.

$\mathbf{P}^0_{\mathbf{acc}}$ **:** If the source of an attack is accepted then its target cannot be accepted.

Using the vocabulary introduced in Section 4.1.1 on the previous page, Property $\mathbf{P}^0_{\mathbf{acc}}$ can be expressed, for any attack (a, b) of the graph, by the conjunction of two first-order formulae:

$\mathbf{P}^0_{\mathbf{acc}}$ For any $(a, b) \in \mathbf{R}$, $(Acc(a) \rightarrow NAcc(b)) \bigwedge (NAcc(b) \rightarrow \neg Acc(b))$

Notes:

- The formula $Acc(a) \rightarrow NAcc(b)$ encodes the attack from a to b (if a is accepted then b is attacked by an accepted argument). It allows for the representation of the direction of the attack and avoids the contraposition of the attack. Indeed, from $\{Acc(a) \rightarrow NAcc(b), NAcc(b) \rightarrow \neg Acc(b)\}$, we can deduce $Acc(a) \rightarrow \neg Acc(b)$, but we cannot deduce $Acc(b) \rightarrow NAcc(a)$.
- The formula $NAcc(b) \rightarrow \neg Acc(b)$ encodes the impact of the attack on b (if b is attacked by an accepted argument, then b is not accepted). It could be equivalently written as $NAcc(b), Acc(b) \rightarrow \bot$ thus removing the connector of negation.
- From $\{Acc(a) \rightarrow NAcc(b), NAcc(b) \rightarrow \neg Acc(b)\}$, we can deduce $Acc(a) \rightarrow \neg Acc(b)$, but not $Acc(b) \rightarrow NAcc(a)$.
- For proving $NAcc(b)$ we need a formula $Acc(x) \rightarrow NAcc(b)$ and a proof for $Acc(x)$.

  So having the symbol $NAcc(x)$ is essential and it cannot be replaced by $\neg Acc(x)$.

The logical base $\Sigma^0(G)$ that describes the graph G consist of the union of the descriptions of all the attacks of G, using Property $\mathbf{P}^0_{\mathbf{acc}}$.
It is easy to prove the consistency of $\Sigma^0(G)$:

Prop. 1 *Let G be an argumentation graph representing an AS with $\mathbf{A}$ the set of arguments and $\mathbf{R} \subseteq \mathbf{A} \times \mathbf{A}$ the set of attacks. The logical knowledge base $\Sigma^0(G)$ is consistent.*

Comparison with the encoding proposed in [14, 35] (see Section 3.2 on page 15) Consider an attack $b\mathbf{R}a$. $NAcc$ plays the same role as def. So we can compare our encoding with the first part of $\Pi(x)$:

- The first part of $\Pi(x)$ gives $\Pi_1 = \{def(a) \leftarrow \neg def(b), acc(a) \leftarrow \neg def(a),\ acc(b) \leftarrow \neg def(b)\}$.
- With Property $\mathbf{P}^0_{\mathbf{acc}}$, we obtain $\{Acc(b) \rightarrow NAcc(a), NAcc(a) \rightarrow \neg Acc(a), NAcc(b) \rightarrow \neg Acc(b)\}$. Then replace $NAcc$ by def and Acc by acc and consider $\Sigma^0 = \{acc(b) \rightarrow def(a), def(a) \rightarrow \neg acc(a), def(b) \rightarrow \neg acc(b)\}$. The sets Σ^0 and Π_1 are not logically equivalent.

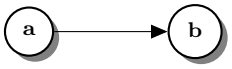
- By completion[2] the rule $acc(x) \leftarrow \neg def(x)$ gives the formula $acc(x) \leftrightarrow \neg def(x)$. Then consider $\Pi'_1 = \{def(a) \leftarrow \neg def(b), acc(a) \leftrightarrow \neg def(a), acc(b) \leftrightarrow \neg def(b)\}$. We can see that Σ^0 is a logical consequence of Π'_1 (but there is no equivalence).
- In the same way, in the case of an AS, the formula $NAcc(a) \rightarrow \neg Acc(a)$ may be completed for obtaining $NAcc(a) \leftrightarrow \neg Acc(a)$. That means that we always have either $Acc(a)$ or $NAcc(a)$. So consider now the set $\Sigma'_0 = \{acc(b) \rightarrow def(a), def(a) \leftrightarrow \neg acc(a), def(b) \leftrightarrow \neg acc(b)\}$. The sets Π'_1 and Σ'_0 are logically equivalent.

4.1.3 Some examples

In this section, some typical examples of AS with simple attacks (*i.e.* no recursive) are encoded, following the definitions of Section 4.1.2 on page 19.

Note: A complete set of examples (including these ones) is given in Appendix A.1 on page 67.

Ex. 1 *In this example, the graph G is reduced to a simple attack from a to b:*



Using $\mathbf{P}^0_{\mathbf{acc}}$ *leads to the following encoding:*

$\Sigma^0(G) = \{$

$Acc(a) \rightarrow NAcc(b)$

$NAcc(b) \rightarrow \neg Acc(b)\}$

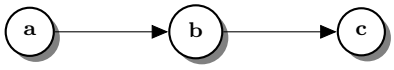
Note that, among the logical consequences of $\Sigma^0(G)$, we find the following formula:

$Acc(a) \rightarrow \neg Acc(b)$

That means that, if a is accepted then b is not accepted.

Ex. 3

In this example, the graph G is reduced to a sequence of two attacks: a attacks b which attacks c.



Using $\mathbf{P}^0_{\mathbf{acc}}$ *leads to the following encoding:*

$\Sigma^0(G) = \{$

$Acc(a) \rightarrow NAcc(b)$

$Acc(b) \rightarrow NAcc(c)$

$NAcc(b) \rightarrow \neg Acc(b)$

[2] *i.e.* we assume that $\neg def(x)$ is the only reason for having $acc(x)$, otherwise there would exist other rules producing $acc(x)$. This completion mechanism is often used when we reason on incomplete knowledge bases.

$NAcc(c) \rightarrow \neg Acc(c)$ }

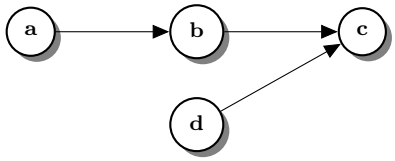
Among the logical consequences of $\Sigma^0(G)$*, we find the following formulae:*

$\neg(Acc(a) \wedge Acc(b))$ *and* $\neg(Acc(c) \wedge Acc(b))$

So, if a *is accepted then* b *is not accepted and* c might be *accepted. But we cannot* deduce *that* c is *accepted since we do not encode the reinstatement principle. We have just encoded the semantics of an attack.*

Ex. 6

In this example, the graph G *contains a defence and an attack to a same argument:*



Using $\mathbf{P}^0_{\mathbf{acc}}$ *leads to the following encoding:*

$\Sigma^0(G) = \{$

$Acc(a) \rightarrow NAcc(b)$

$Acc(b) \rightarrow NAcc(c)$

$Acc(d) \rightarrow NAcc(c)$

$NAcc(b) \rightarrow \neg Acc(b)$

$NAcc(c) \rightarrow \neg Acc(c)$ }

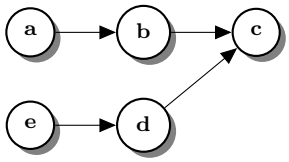
Among the logical consequences of $\Sigma^0(G)$*, we find the following formulae:*

$\neg(Acc(a) \wedge Acc(b))$, $\neg(Acc(b) \wedge Acc(c))$ *and* $\neg(Acc(d) \wedge Acc(c))$

So, if a *is accepted then* b *is not accepted, and, if* b *or* d *is accepted then* c *is not accepted.*

Ex. 9

In this example, the graph G *contains two defences against two distinct attackers for a same argument:*



Using $\mathbf{P}^0_{\mathbf{acc}}$ *leads to the following encoding:*

$\Sigma^0(G) = \{$

$Acc(a) \rightarrow NAcc(b)$

$Acc(b) \rightarrow NAcc(c)$

$Acc(e) \rightarrow NAcc(d)$

$Acc(d) \rightarrow NAcc(c)$

$NAcc(b) \rightarrow \neg Acc(b)$

$NAcc(c) \rightarrow \neg Acc(c)$

$NAcc(d) \rightarrow \neg Acc(d)$ }

Among the logical consequences of $\Sigma^0(G)$, we find the following formulae:

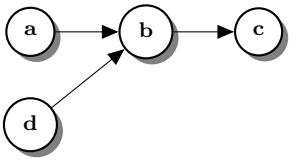
$\neg(Acc(a) \wedge Acc(b))$, $\neg(Acc(b) \wedge Acc(c))$, $\neg(Acc(e) \wedge Acc(d))$ *and* $\neg(Acc(d) \wedge Acc(c))$

So, if a (resp. b, e and d) is accepted, b (resp. c, d and c) is not accepted.

Note that, if a ***and*** *e are accepted, c* might be *accepted. But it is not deductible from $\Sigma^0(G)$.*

Ex. 10

In this example, the graph G contains two defences against a same attacker:



Using $\mathbf{P}^0_{\mathbf{acc}}$ leads to the following encoding:

$\Sigma^0(G) = \{$

$Acc(a) \rightarrow NAcc(b)$

$Acc(b) \rightarrow NAcc(c)$

$Acc(d) \rightarrow NAcc(b)$

$NAcc(b) \rightarrow \neg Acc(b)$

$NAcc(c) \rightarrow \neg Acc(c)$ }

Among the logical consequences of $\Sigma^0(G)$, we find the following formulae:

$\neg(Acc(a) \wedge Acc(b))$, $\neg(Acc(b) \wedge Acc(c))$ *and* $\neg(Acc(d) \wedge Acc(b))$

So, if a (resp. b and d) is accepted, b (resp. c and b) is not accepted.

Note that, if a ***or*** *d is accepted, c* might be *accepted. But it is not deductible from $\Sigma^0(G)$.*

Other examples are available in Appendix A.1 on page 67.

4.2 Extended language for an explicit representation of attacks (recursive case)

Our aim is to describe an argumentation graph G with an explicit representation of the attacks, that might be themselves attacked. This description must allow us to represent arguments, attacks and their properties (accepted argument, attacked argument, grounded attack, valid attack, ...). $\Sigma(G)$ will denote the set of logical formulae describing the argumentation graph G.

Our idea is to separate the formulas that encode the properties of an attack in the general case from the formulas that describe a particular argumentation graph. For that purpose, we need first-order

logic (with the classical connectors $\vee$, $\wedge$, $\rightarrow$, $\leftrightarrow$, $\neg$, and the quantifiers $\exists$ and $\forall$). As we have to distinguish between arguments and attacks, we will use bounded quantifiers.[3]

4.2.1 Vocabulary

We use the following unary predicate symbols : Acc, $NAcc$, Gr, Val, Act, $Attack$, $Argument$ 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
- $Gr(\alpha)$ (resp. $Val(\alpha)$, $Act(\alpha)$) means "α is grounded" (resp. "α is valid, "α is active") when α denotes an attack
- $Attack(x)$ means "x is an attack"
- $Argument(x)$ means "x is an argument"
- $T(x)$ (resp. $S(x)$) denotes the target (resp. source) of x, when x denotes an attack

We will also use the binary equality predicate.

Note that the quantifiers $\exists$ and $\forall$ range over some domain D. To restrict them to subsets of D, bounded quantifiers will be also used:

$(\forall x \in E)(P(x))$ means $(\forall x)(x \in E \rightarrow P(x))$ or equivalently $(\forall x)(E(x) \rightarrow P(x))$.

Here we will use:

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

Notes:

- As shown below by Property $\mathbf{P_{act}}$, $Act(\alpha)$ is not really mandatory, but rather enables us to simplify the formulae.
- As in Section 4.1 on page 19, the meaning of $NAcc(x)$ is not "x is not accepted" but rather "x cannot be accepted". In other words, due to the fact that attacks may be attacked, the meaning of $NAcc(x)$ is "x is the target of a valid attack whose source is accepted".

4.2.2 Properties

The properties introduced in this section can be partitioned into two sets:

- the properties describing the general behaviour of an attack, possibly recursive, in an argumentation graph, *i.e.* how an attack interacts with arguments and other attacks related to it.
- and the properties encoding the specificities of the current argumentation graph.

[3] It was not necessary to use bounded quantifiers in Section 4.1 on page 19, since in this section, the variables in formulae could only denote arguments.

General theory. Regarding an attack, we first define the notion of grounded and active attack from the notion of valid attack, with Properties $\mathbf{P_{gr}}$ and $\mathbf{P_{act}}$.

Then, the behaviour of an attack wrt its source and its target can be expressed using the 2 following properties $\mathbf{P_{val}}$ and $\mathbf{P_{acc}}$. These properties generalize Property $\mathbf{P^0_{acc}}$ introduced in Section 4.1.2 on page 19. The idea is to describe the semantics of an attack under the form of a constraint on its source (an argument) and its target (an attack or an argument).

$\mathbf{P_{gr}}$ **:** An attack is grounded iff its source is an accepted argument.

$\mathbf{P_{act}}$ **:** An attack is active iff it is grounded and valid.

$\mathbf{P_{acc}}$ **:** If an attack between two arguments is valid, then if its source is accepted, its target cannot be accepted.

$\mathbf{P_{val}}$ **:** If an attack from an argument to an attack is valid, then if its source is accepted, its target cannot be valid.

An equivalent formulation of $\mathbf{P_{val}}$ is :
If an attack is the target of an active attack, then it is not valid.

Notes:

- $\mathbf{P_{acc}}$ is an extended version of $\mathbf{P^0_{acc}}$
- Due to Properties $\mathbf{P_{gr}}$ and $\mathbf{P_{act}}$, the notions of grounded and active attack can be expressed using the notions of accepted argument and valid attack. That will enable us to simplify the language.
- The above properties do not give a definition of a valid attack. Property $\mathbf{P_{val}}$ only gives a sufficient condition for an attack being not valid.

Using the vocabulary defined in Section 4.2.1 on the preceding page, the above properties can be expressed by the following set of first-order formulae, denoted by Π (this set is always the same regardless of the processed graph):

(1) $\forall x \in Attack(Gr(x) \leftrightarrow Acc(S(x)))$

(2) $\forall x \in Attack(Act(x) \leftrightarrow (Gr(x) \wedge Val(x)))$

(3) $\forall x \in Attack(\forall y \in Attack(Val(y) \wedge (T(y) = x) \wedge Acc(S(y)) \rightarrow \neg Val(x))$

(4) $\forall x \in Argument(\forall y \in Attack(Val(y) \wedge (T(y) = x) \wedge Acc(S(y)) \rightarrow NAcc(x))$

(5) $\forall x \in Argument(NAcc(x) \rightarrow \neg Acc(x))$

(6) $\forall x(Attack(x) \rightarrow \neg Argument(x))$

(7) $\forall x(Argument(x) \vee Attack(x))$

Formulae **(1, 2)** express the properties $\mathbf{P_{gr}}, \mathbf{P_{act}}$. Formula **(3)** expresses Property $\mathbf{P_{val}}$. Formulae **(4-5)** express Property $\mathbf{P_{acc}}$. Formula **(6)** says that arguments are not attacks and attacks are not arguments, and Formula **(7)** prevents the language from talking about anything else than attacks and arguments.

Logical encoding of specificities of G. Let G be an argumentation graph representing an ASAF with $\mathbf{A}$ the set of arguments and $\mathbf{R}$ the set of attacks. We assume that G is finite with $\mathbf{A} = \{a_1, \ldots, a_n\}$ and the set of interactions is $\{\alpha_1, \ldots, \alpha_m\}$. In order to make arguments a and b appear when the attack $\alpha = (a, b)$ is encoded, we consider the literals $(S(\alpha) = a)$ and $(T(\alpha) = b)$.[4]

Let $\Pi(G)$ denote the following set of formulas:

(8) $(S(\alpha) = a) \wedge (T(\alpha) = b)$ for all $\alpha \in \mathbf{R}$ with $\alpha = (a, b)$

(9) $\forall x(Argument(x) \leftrightarrow (x = a_1) \vee \ldots \vee (x = a_n))$

(10) $\forall x(Attack(x) \leftrightarrow (x = \alpha_1) \vee \ldots \vee (x = \alpha_m))$

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

(12) $\alpha_i \neq \alpha_j$ for all $\alpha_i, \alpha_j \in \mathbf{R}$ with $i \neq j$

The logical theory $\Sigma(G)$ corresponding to the argumentation graph G consists of the union of the theories Π and $\Pi(G)$.

Note that formulae **(7, 9, 10)** force the interpretation of terms to be surjective and thus models of $\Sigma(G)$ satisfy the Domain Closure Assumption (DCA).

Moreover, Formula **(6)** together with formulae **(9, 10)** imply:

(13) $a_i \neq \alpha_j$ for all $a_i \in \mathbf{A}$ and $\alpha_j \in \mathbf{R}$

And Formula **(13)** together with formulae **(11,12)** establish the Unique Name Assumption (UNA) for all constants of the language (though not for functional terms so that Formula **(8)** is not a contradiction).

Note also that the terms $S(\alpha)$ and $T(\alpha)$ are defined for every attack $\alpha \in \mathbf{R}$, but they are not defined for arguments so they can take any value (we will not be interested in the values they take for arguments).

Notes about $\mathbf{P_{val}}$: Let us consider the following examples:

- In the particular case of an AS with simple attacks (no attacked attack), Formula **(3)** is a tautology (since there exists no attack that attacks another attack) and thus can be ignored. Moreover, as Π does not give any sufficient condition for an attack being valid, it cannot be assumed that the attacks in the AS are valid. As a consequence, in order to recover Dung's semantics in the case of an AS, it will be necessary to add a specific property (see Section 5.2.1 on page 44).
- Consider now a graph G reduced to one argument a and one attack α such that $S(\alpha) = a$ and $T(\alpha) = \alpha$. Due to Property $\mathbf{P_{val}}$, $\Sigma(G)$ enables to deduce the formula : $Val(\alpha) \wedge Acc(a) \rightarrow \neg Val(\alpha)$ which is equivalent to the formula : $Acc(a) \rightarrow \neg Val(\alpha)$. So neither $Val(\alpha)$, nor $\neg Val(\alpha)$ can be deduced.

[4] In the examples we will write S_α in place of $S(\alpha)$ and T_α in place of $T(\alpha)$ for simplicity.

It is easy to prove the consistency of $\Sigma(G)$:

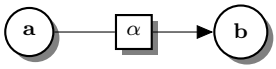
Prop. 2 *Let G be an argumentation graph representing an ASAF with $\mathbf{A}$ the set of arguments and $\mathbf{R}$ the set of attacks. The logical knowledge base $\Sigma(G)$ is consistent.*

4.2.3 Some examples

In this section, we revisit the examples given in Section 4.1.3 on page 21 and add some new examples presenting recursive attacks. For each example, we give its logical description (a complete description for Example 1, then a simplified one for the other examples).

Note: A more complete set of examples is given in appendices A.1 on page 67 and A.2 on page 91.

Ex. 1 (cont'd) *In this example, the graph G is reduced to a simple attack: $\alpha = (a, b)$*



Π *is given as in Section 4.2.2 on page 24 and $\Pi(G)$ contains the following formulas:*

$(S(\alpha) = a) \wedge (T(\alpha) = b)$ *(the simplified form being $(s_\alpha = a) \wedge (t_\alpha = b)$)*

$\forall x(Argument(x) \leftrightarrow (x = a) \vee (x = b))$

$\forall x(Attack(x) \leftrightarrow (x = \alpha)$

$a \neq b$

So from $\Sigma(G) = \Pi \cup \Pi(G)$, we can conclude the formulas $a \neq \alpha$ and $b \neq \alpha$.

Moreover, using the equality axioms, a simplified version of $\Sigma(G)$ can be obtained (in particular, the tautologies are not given), which is easy-to-read:[5]

$\Sigma(G) = \Pi \cup \Pi(G) = \{$

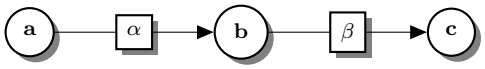
$Gr(\alpha) \leftrightarrow Acc(a)$ *(*$\mathbf{P_{gr}}$*)*

$Act(\alpha) \leftrightarrow (Gr(\alpha) \wedge Val(\alpha))$ *(*$\mathbf{P_{act}}$*)*

$(Val(\alpha) \wedge Acc(a)) \rightarrow NAcc(b)$ *and* $NAcc(b) \rightarrow \neg Acc(b)$ *(*$\mathbf{P_{acc}}$*)*$\}$

Note that we can deduce from $\Sigma(G)$ neither $Val(\alpha)$, nor $Acc(a)$. However, it can be deduced that $(Val(\alpha) \wedge Acc(a)) \rightarrow \neg Acc(b)$.

Ex. 3 (cont'd) *In this example, the graph G is reduced to a sequence of two attacks: $\alpha = (a, b)$ and $\beta = (b, c)$*



[5] On the other examples, we will give directly the simplified version of $\Sigma(G)$.

Let $s_\alpha = a$, $t_\alpha = b$, $s_\beta = b$, $t_\beta = c$.

Properties are instantiated on the domain defined by G *(three arguments* a, b *and* c *and two attacks* α *and* β*).*

$\Sigma(G) = \{$

$Gr(\alpha) \leftrightarrow Acc(a)$ (**P**$_{\mathbf{gr}}$)

$Gr(\beta) \leftrightarrow Acc(b)$ (**P**$_{\mathbf{gr}}$)

$Act(\alpha) \leftrightarrow (Gr(\alpha) \wedge Val(\alpha))$ (**P**$_{\mathbf{act}}$)

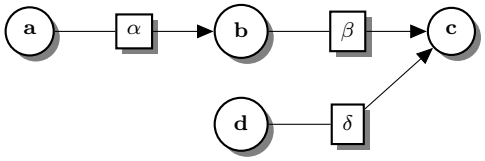
$Act(\beta) \leftrightarrow (Gr(\beta) \wedge Val(\beta))$ (**P**$_{\mathbf{act}}$)

$(Val(\alpha) \wedge Acc(a)) \rightarrow NAcc(b)$ *and* $NAcc(b) \rightarrow \neg Acc(b)$ (**P**$_{\mathbf{acc}}$)

$(Val(\beta) \wedge Acc(b)) \rightarrow NAcc(c)$ *and* $NAcc(c) \rightarrow \neg Acc(c)$ (**P**$_{\mathbf{acc}}$) $\}$

Ex. 6 (cont'd)

In this example, the graph G *contains a defence and an attack to a same argument:* $\alpha = (a, b)$, $\beta = (b, c)$ *and* $\delta = (d, c)$*:*



Let $s_\alpha = a$, $t_\alpha = b$, $s_\beta = b$, $t_\beta = c$, $s_\delta = d$, $t_\delta = c$.

Properties are instantiated on the domain defined by G *(four arguments* a, b, c *and* d *and three attacks* α, β *and* δ*).*

$\Sigma(G) = \{$

$Gr(\alpha) \leftrightarrow Acc(a)$ (**P**$_{\mathbf{gr}}$)

$Gr(\beta) \leftrightarrow Acc(b)$ (**P**$_{\mathbf{gr}}$)

$Gr(\delta) \leftrightarrow Acc(d)$ (**P**$_{\mathbf{gr}}$)

$Act(\alpha) \leftrightarrow (Gr(\alpha) \wedge Val(\alpha))$ (**P**$_{\mathbf{act}}$)

$Act(\beta) \leftrightarrow (Gr(\beta) \wedge Val(\beta))$ (**P**$_{\mathbf{act}}$)

$Act(\delta) \leftrightarrow (Gr(\delta) \wedge Val(\delta))$ (**P**$_{\mathbf{act}}$)

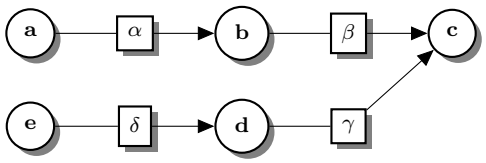
$(Val(\alpha) \wedge Acc(a)) \rightarrow NAcc(b)$ *and* $NAcc(b) \rightarrow \neg Acc(b)$ (**P**$_{\mathbf{acc}}$)

$(Val(\beta) \wedge Acc(b)) \rightarrow NAcc(c)$ *and* $NAcc(c) \rightarrow \neg Acc(c)$ (**P**$_{\mathbf{acc}}$)

$(Val(\delta) \wedge Acc(d)) \rightarrow NAcc(c)$ *and* $NAcc(c) \rightarrow \neg Acc(c)$ (**P**$_{\mathbf{acc}}$) $\}$

Ex. 9 (cont'd)

In this example, the graph G *contains two defences against two distinct attackers for a same argument:* $\alpha = (a, b)$, $\beta = (b, c)$, $\delta = (e, d)$ *and* $\gamma = (d, c)$*:*



Let $s_\alpha = a$, $t_\alpha = b$, $s_\beta = b$, $t_\beta = c$, $s_\delta = e$, $t_\delta = d$, $s_\gamma = d$, $t_\gamma = c$.

Properties are instantiated on the domain defined by G *(five arguments* a, b, c, d *and* e *and four attacks* α, β, δ *and* γ*).*

$\Sigma(G) = \{$

$Gr(\alpha) \leftrightarrow Acc(a)$ **(P$_{\text{gr}}$)**

$Gr(\beta) \leftrightarrow Acc(b)$ **(P$_{\text{gr}}$)**

$Gr(\delta) \leftrightarrow Acc(e)$ **(P$_{\text{gr}}$)**

$Gr(\gamma) \leftrightarrow Acc(d)$ **(P$_{\text{gr}}$)**

$Act(\alpha) \leftrightarrow (Gr(\alpha) \wedge Val(\alpha))$ **(P$_{\text{act}}$)**

$Act(\beta) \leftrightarrow (Gr(\beta) \wedge Val(\beta))$ **(P$_{\text{act}}$)**

$Act(\delta) \leftrightarrow (Gr(\delta) \wedge Val(\delta))$ **(P$_{\text{act}}$)**

$Act(\gamma) \leftrightarrow (Gr(\gamma) \wedge Val(\gamma))$ **(P$_{\text{act}}$)**

$(Val(\alpha) \wedge Acc(a)) \rightarrow NAcc(b)$ *and* $NAcc(b) \rightarrow \neg Acc(b)$ **(P$_{\text{acc}}$)**

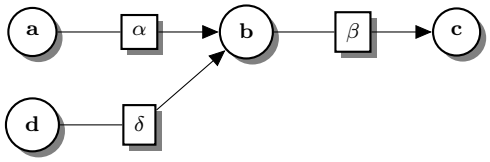
$(Val(\beta) \wedge Acc(b)) \rightarrow NAcc(c)$ *and* $NAcc(c) \rightarrow \neg Acc(c)$ **(P$_{\text{acc}}$)**

$(Val(\delta) \wedge Acc(e)) \rightarrow NAcc(d)$ *and* $NAcc(d) \rightarrow \neg Acc(d)$ **(P$_{\text{acc}}$)**

$(Val(\gamma) \wedge Acc(d)) \rightarrow NAcc(c)$ *and* $NAcc(c) \rightarrow \neg Acc(c)$ **(P$_{\text{acc}}$)** $\}$

Ex. 10 (cont'd)

In this example, the graph G *contains two defences against a same attacker:* $\alpha = (a, b)$, $\beta = (b, c)$ *and* $\delta = (d, b)$*:*



Let $s_\alpha = a$, $t_\alpha = b$, $s_\beta = b$, $t_\beta = c$, $s_\delta = d$, $t_\delta = b$.

Properties are instantiated on the domain defined by G *(four arguments* a, b, c *and* d *and three attacks* α, β *and* δ*).*

$\Sigma(G) = \{$

$Gr(\alpha) \leftrightarrow Acc(a)$ **(P$_{\text{gr}}$)**

$Gr(\beta) \leftrightarrow Acc(b)$ **(P$_{\text{gr}}$)**

$Gr(\delta) \leftrightarrow Acc(d)$ **(P$_{\text{gr}}$)**

$Act(\alpha) \leftrightarrow (Gr(\alpha) \wedge Val(\alpha))$ **(P$_{\text{act}}$)**

$Act(\beta) \leftrightarrow (Gr(\beta) \wedge Val(\beta))$ **(P$_{\text{act}}$)**

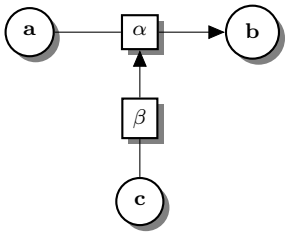
$Act(\delta) \leftrightarrow (Gr(\delta) \wedge Val(\delta))$ **(P$_{\text{act}}$)**

$(Val(\alpha) \wedge Acc(a)) \rightarrow NAcc(b)$ *and* $NAcc(b) \rightarrow \neg Acc(b)$ **(P$_{\text{acc}}$)**

$(Val(\beta) \wedge Acc(b)) \rightarrow NAcc(c)$ *and* $NAcc(c) \rightarrow \neg Acc(c)$ **(P$_{\text{acc}}$)**

$(Val(\delta) \wedge Acc(d)) \rightarrow NAcc(b)$ *and* $NAcc(b) \rightarrow \neg Acc(b)$ **(P$_{\text{acc}}$)** $\}$

Ex. 14 *In this example, the graph* G *is reduced to an attacked attack:* $\alpha = (a, b)$, $\beta = (c, \alpha)$



Let $s_\alpha = a$, $t_\alpha = b$, $s_\beta = c$, $t_\beta = \alpha$.

Properties are instantiated on the domain defined by G *(three arguments* a, b, c *and two attacks* α *and* β*).*

$\Sigma(G) = \{$

$Gr(\alpha) \leftrightarrow Acc(a)$ **(P$_{\mathbf{gr}}$)**

$Gr(\beta) \leftrightarrow Acc(c)$ **(P$_{\mathbf{gr}}$)**

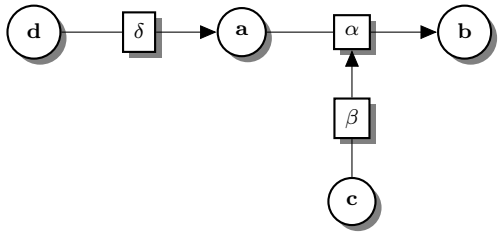
$(Val(\beta) \wedge Acc(c)) \rightarrow \neg Val(\alpha))$ **(P$_{\mathbf{val}}$)**

$Act(\alpha) \leftrightarrow (Gr(\alpha) \wedge Val(\alpha))$ **(P$_{\mathbf{act}}$)**

$Act(\beta) \leftrightarrow (Gr(\beta) \wedge Val(\beta))$ **(P$_{\mathbf{act}}$)**

$(Val(\alpha) \wedge Acc(a)) \rightarrow NAcc(b)$ *and* $NAcc(b) \rightarrow \neg Acc(b)$ **(P$_{\mathbf{acc}}$)** $\}$

Ex. 17



Let $s_\alpha = a$, $t_\alpha = b$, $s_\beta = c$, $t_\beta = \alpha$, $s_\delta = d$, $t_\delta = a$.

Properties are instantiated on the domain defined by G *(four arguments* a, b, c, d *and three attacks* α, β *and* δ*).*

$\Sigma(G) = \{$

$Gr(\alpha) \leftrightarrow Acc(a)$ **(P$_{\mathbf{gr}}$)**

$Gr(\beta) \leftrightarrow Acc(c)$ **(P$_{\mathbf{gr}}$)**

$Gr(\delta) \leftrightarrow Acc(d)$ **(P$_{\mathbf{gr}}$)**

$(Val(\beta) \wedge Acc(c)) \rightarrow \neg Val(\alpha))$ **(P$_{\mathbf{val}}$)**

$Act(\alpha) \leftrightarrow (Gr(\alpha) \wedge Val(\alpha))$ **(P$_{\mathbf{act}}$)**

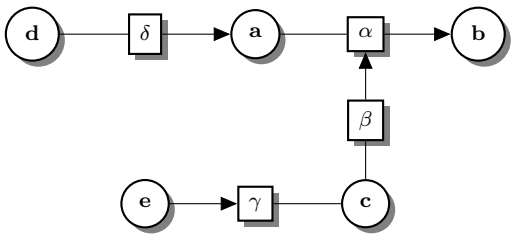
$Act(\beta) \leftrightarrow (Gr(\beta) \wedge Val(\beta))$ **(P$_{\mathbf{act}}$)**

$Act(\delta) \leftrightarrow (Gr(\delta) \wedge Val(\delta))$ **(P$_{\mathbf{act}}$)**

$(Val(\alpha) \wedge Acc(a)) \rightarrow NAcc(b)$ *and* $NAcc(b) \rightarrow \neg Acc(b)$ **(P$_{\mathbf{acc}}$)**

$Acc(d) \rightarrow NAcc(a)$ *and* $NAcc(a) \rightarrow \neg Acc(a)$ $\}$ **(P$_{\mathbf{acc}}$)**

Ex. 18



Let $s_\alpha = a$, $t_\alpha = b$, $s_\beta = c$, $t_\beta = \alpha$, $s_\delta = d$, $t_\delta = a$, $s_\gamma = e$, $t_\gamma = c$.

Properties are instantiated on the domain defined by G *(five arguments* a, b, c, d, e *and four attacks* α, β, δ *and* γ*).*

$\Sigma(G) = \{$

$Gr(\alpha) \leftrightarrow Acc(a)$ *(*$\mathbf{P_{gr}}$*)*

$Gr(\beta) \leftrightarrow Acc(c)$ *(*$\mathbf{P_{gr}}$*)*

$Gr(\delta) \leftrightarrow Acc(d)$ *(*$\mathbf{P_{gr}}$*)*

$Gr(\gamma) \leftrightarrow Acc(e)$ *(*$\mathbf{P_{gr}}$*)*

$(Val(\beta) \wedge Acc(c)) \rightarrow \neg Val(\alpha))$ *(*$\mathbf{P_{val}}$*)*

$Act(\alpha) \leftrightarrow (Gr(\alpha) \wedge Val(\alpha))$ *(*$\mathbf{P_{act}}$*)*

$Act(\beta) \leftrightarrow (Gr(\beta) \wedge Val(\beta))$ *(*$\mathbf{P_{act}}$*)*

$Act(\delta) \leftrightarrow (Gr(\delta) \wedge Val(\delta))$ *(*$\mathbf{P_{act}}$*)*

$Act(\gamma) \leftrightarrow (Gr(\gamma) \wedge Val(\gamma))$ *(*$\mathbf{P_{act}}$*)*

$(Val(\alpha) \wedge Acc(a)) \rightarrow NAcc(b)$ *and* $NAcc(b) \rightarrow \neg Acc(b)$ *(*$\mathbf{P_{acc}}$*)*

$Acc(d) \rightarrow NAcc(a)$ *and* $NAcc(a) \rightarrow \neg Acc(a)$ *(*$\mathbf{P_{acc}}$*)*

$Acc(e) \rightarrow NAcc(c)$ *and* $NAcc(c) \rightarrow \neg Acc(c)$ $\}$ *(*$\mathbf{P_{acc}}$*)*

Many other examples are available in Appendix A on page 67.

Chapter 5

Logical formalization of semantics: Case of AS

In argumentation theory, an extension-based semantics is defined by a set of requirements that a set of arguments should satisfy. In this section, our purpose is to restate standard semantics of an AS in logical terms.

The first step is to consider the case of an AS represented in the basic language (see Section 5.1) and characterize standard semantics by logical formulas of this basic language. Next we will consider the extended version of the language (see Section 5.2 on page 44) allowing for the explicit representation of attacks, and show that the logical formalization of standard semantics of an AS can be easily obtained.

Building on this logical formalization of an AS in the extended language, we will be able to propose semantics accounting for recursive attacks in Section 6 on page 51.

Note that ideas and results obtained in Section 5.1 correspond to those introduced in [8], but with a different and richer language that allows us to extend our work to recursive attacks.

5.1 Semantics for an AS in the basic language

In this section, the base corresponding to the translation of the argumentation graph G is Σ^0.

As recalled in Section 2.2 on page 7, standard semantics are based on three basic principles: the conflict-freeness principle, the defence principle (closely related to the notion of admissibility), and its dual, the reinstatement principle.

We successively consider these three principles, giving a logical expression for each of them.

5.1.1 Conflict-freeness

Each semantics demands that its extensions be conflict-free. This requirement can be formulated as follows:

$\mathbf{P_{cf}}$ If two arguments are conflicting, they cannot be jointly accepted.

The attack $a\mathbf{R}b$ is encoded in the base $\Sigma^0(G)$ by the formula $Acc(a) \rightarrow NAcc(b)$. So $S \subseteq A$ is not conflict-free iff there exists in $\Sigma^0(G)$ a formula $Acc(a) \rightarrow NAcc(b)$ with $a, b \in S$. On the other hand, the formula $Acc(a) \rightarrow NAcc(b)$ comes with the formula $NAcc(b) \rightarrow \neg Acc(b)$. So the base $\Sigma^0(G)$ already contains formulas that express the property $\mathbf{P_{cf}}$.

5.1.2 Defence

As recalled in Section 2.2 on page 7, an argument a is acceptable wrt a set of arguments S if S attacks each argument attacker of a. As an immediate consequence, we have that an argument which is not attacked is always acceptable wrt S. Moreover, if a set of arguments S is admissible, it defends all its elements. So, if an argument a belongs to an admissible set S, either a is unattacked, or a is defended by S against each of its attackers.

These remarks lead to formulate the following property $\mathbf{P_{def}}$ for describing the defence.

$\mathbf{P_{def}}$: An "attacked" argument may be accepted only if it is defended by an accepted argument against each of its attackers.

Let us enumerate interesting particular cases:

1. Case of a sequence of two attacks: a attacks b that attacks c. c may be accepted only if a is accepted, as a is the unique defender of c against b. This constraint results in the formula $Acc(c) \rightarrow Acc(a)$.

2. Case of several defenders against the same attacker: $b\mathbf{R}c$ and $a_1\mathbf{R}b, \ldots a_k\mathbf{R}b$ (the attackers of b are $a_1, \ldots a_k$). c may be accepted only if one of the a_i is accepted. This constraint results in the formula $Acc(c) \rightarrow (Acc(a_1) \vee \ldots \vee Acc(a_k))$.

3. Case of a single attacker and no defender: b attacks c and there is no attack to b. c may be accepted only if one of the attackers of b is accepted. Yet the set of attackers of b is empty and the disjunction of an empty set of literals is always *false*. So we obtain the formula $Acc(c) \rightarrow \bot$ which is equivalent to $\neg Acc(c)$. It may be noted that there is an analogy with the logical encoding proposed by [14, 35] (see Section 3.2 on page 15).

4. Case of several attackers: $b_1\mathbf{R}c, \ldots b_k\mathbf{R}c$ and $\forall i, a_i\mathbf{R}b_i$. c may be accepted only if it is defended against each of its attackers. So each a_i must be accepted. This constraint results in the formula $Acc(c) \rightarrow (Acc(a_1) \wedge \ldots \wedge Acc(a_k))$.

The particular cases considered above lead to the formula that expresses Property $\mathbf{P_{def}}$.

Let c be an attacked argument and $\mathbf{R}^-(c) = \{b_1, \ldots b_k\}$ its attackers, for each $b_i \in \mathbf{R}^-(c)$, $B_i = \mathbf{R}^-(b_i)$ denotes the set (possibly empty) of the arguments that attack b_i. The property $\mathbf{P_{def}}$ for c is expressed by the formula $Acc(c) \rightarrow (\wedge_{i=1..k}(\vee_{a \in B_i} Acc(a))))$. More generally and formally, Property $\mathbf{P_{def}}$ is expressed by:

$\mathbf{P_{def}}$ $(\forall c)(Acc(c) \rightarrow (\wedge_{b \in \mathbf{R}^-(c)}(\vee_{a \in \mathbf{R}^-(b)} Acc(a))))$

Note that the formula $Acc(c) \rightarrow (\wedge_{b \in \mathbf{R}^-(c)}(\vee_{a \in \mathbf{R}^-(b)} Acc(a)))$ is logically equivalent to the conjunction of the formulas $Acc(c) \rightarrow (\vee_{a \in \mathbf{R}^-(b)} Acc(a))$, $b \in \mathbf{R}^-(c)$. That is to say that $\mathbf{P}_{\mathbf{def}}$ results in one formula for each attack.

Remark
We could have removed the word "attacked" in the formulation of Property $\mathbf{P}_{\mathbf{def}}$, instead writing "An argument may be accepted only if it is defended by an accepted argument against each of its attackers ". That does not correspond to admissibility, as recalled above. Knowing that an argument is acceptable wrt S implies that "if it is attacked, then ...". Anyway, if this new version was taken, in the particular case of an unattacked argument c, we would get the formula $Acc(c) \rightarrow \top$ (the conjunction of an empty set of literals is always *true*) which is a tautology. The case of an unattacked argument will be relevant for the reinstatement principle.

The defence principle can be explicitely encoded by adding to the base $\Sigma^0(G)$ (which describes the argumentation graph G) the formulas issued from $\mathbf{P}_{\mathbf{def}}$, one formula for each attack to each attacked argument. Let $\Sigma^0_d(G)$ denote the resulting base.

The logical consistency of $\Sigma^0_d(G)$ can be easily verified:

Prop. 3 *Let G be an argumentation graph representing an AS with $\mathbf{A}$ the set of arguments and $\mathbf{R} \subseteq \mathbf{A} \times \mathbf{A}$ the set of attacks. The logical knowledge base $\Sigma^0_d(G)$ is consistent.*

Let us illustrate $\Sigma^0_d(G)$ on different examples.

Ex. 1 (cont'd) *The formula $\neg Acc(b)$ is added to $\Sigma^0(G)$*

Ex. 3 (cont'd) *The formulae $\neg Acc(b)$ and $Acc(c) \rightarrow Acc(a)$ are added to $\Sigma^0(G)$*

Ex. 6 (cont'd) *The formulae $\neg Acc(b)$ et $\neg Acc(c)$ are added to $\Sigma^0(G)$*

Ex. 9 (cont'd) *The formulae $\neg Acc(d)$, $\neg Acc(b)$ and $Acc(c) \rightarrow (Acc(a) \wedge Acc(e))$ are added to $\Sigma^0(G)$*

Ex. 10 (cont'd) *The formulae $\neg Acc(b)$ and $Acc(c) \rightarrow (Acc(a) \vee Acc(d))$ are added to $\Sigma^0(G)$*

5.1.3 Reinstatement

As recalled in Section 2.2 on page 7, if a semantics satisfies the reinstatement principle, then, if an extension S defends an argument a (that is a is acceptable wrt S), a must belong to S. As an unattacked argument is always acceptable wrt S, it is defended by S.

These remarks lead to formulate the following property $\mathbf{P}_{\mathbf{reins}}$ for describing the notion of reinstatement.

$\mathbf{P}_{\mathbf{reins}}$: If an argument is defended by an accepted argument against each of its attackers, then it must be accepted.

Let us enumerate interesting particular cases:

1. Case of a sequence of two attacks: a attacks b that attacks c. If a is accepted, then c must be accepted ("reinstated" by a). This constraint results in the formula $Acc(a) \rightarrow Acc(c)$.

2. Case of several defenders against the same attacker: $b\mathbf{R}c$ and $a_1\mathbf{R}b, \ldots a_k\mathbf{R}b$ (the attackers of b are $a_1, \ldots a_k$). If one of the a_i is accepted then c must be accepted. This constraint results in the formula $(Acc(a_1) \vee \ldots \vee Acc(a_k)) \rightarrow Acc(c)$.

3. Case of several attackers : $b_1\mathbf{R}c, \ldots b_k\mathbf{R}c$ and $\forall i, a_i\mathbf{R}b_i$. If each a_i is accepted, then c must be accepted. This constraint results in the formula $(Acc(a_1) \wedge \ldots \wedge Acc(a_k)) \rightarrow Acc(c)$.

4. Case of a single attacker and no defender: b attacks c and there is no attack to b. The set of attackers of b is empty. We obtain the formula $\bot \rightarrow Acc(c)$ which is a tautology.

5. Case of an unattacked argument: If c is not attacked, it does not have to be defended, it is obviously acceptable. We obtain the formula $\top \rightarrow Acc(c)$ (the conjunction of an empty set of literals is always *true*) equivalent to $Acc(c)$.

The particular cases considered above lead to the formula that expresses Property $\mathbf{P}_{\textbf{reins}}$.

Let c be an argument and $\mathbf{R}^-(c) = \{b_1, \ldots b_k\}$ its attackers, for each $b_i \in \mathbf{R}^-(c)$, $B_i = \mathbf{R}^-(b_i)$ denotes the set (possibly empty) of the arguments that attack b_i. The property $\mathbf{P}_{\textbf{reins}}$ for c is expressed by the formula $(\wedge_{i=1..k}(\vee_{a\in B_i} Acc(a))) \rightarrow Acc(c)$.

More generally and formally, Property $\mathbf{P}_{\textbf{reins}}$ is expressed by:

$\mathbf{P}_{\textbf{reins}}$ $(\forall c)((\wedge_{b\in\mathbf{R}^-(c)}(\vee_{a\in\mathbf{R}^-(b)} Acc(a))) \rightarrow Acc(c))$

Remark: In the particular case of an unattacked argument c, B_i is empty. As the conjunction of an empty set of literals is always *true*, $\mathbf{P}_{\textbf{reins}}$ results in the formula $Acc(c)$.

The reinstatement principle can be explicitely encoded by adding to the base $\Sigma^0(G)$ (which describes the argumentation graph G) the formulas issued from $\mathbf{P}_{\textbf{reins}}$, one formula for each argument (either attacked or not). Let $\Sigma^0_r(G)$ denote the resulting base.

The base $\Sigma^0_r(G)$ is logically consistent:

Prop. 4 *Let G be an argumentation graph representing an AS with $\mathbf{A}$ the set of arguments and $\mathbf{R} \subseteq \mathbf{A} \times \mathbf{A}$ the set of attacks. The logical knowledge base $\Sigma^0_r(G)$ is consistent.*

Let us illustrate $\Sigma^0_r(G)$ on different examples.

Ex. 1 (cont'd) *a is not attacked: the formula $Acc(a)$ is added to $\Sigma^0(G)$. Then, $\neg Acc(b)$ is deducible from $\Sigma^0_r(G)$.*

Ex. 3 (cont'd) *a is not attacked: the formula $Acc(a)$ is added to $\Sigma^0(G)$.*
c is attacked and defended: the formula $Acc(a) \rightarrow Acc(c)$ is added to $\Sigma^0(G)$. The formulae $Acc(c)$ et $\neg Acc(b)$ are deducible from $\Sigma^0_r(G)$.

Ex. 6 (cont'd) *a are d are unattacked: the formulae $Acc(a)$ et $Acc(d)$ are added to $\Sigma^0(G)$.*
c is attacked with two attackers: the formula $(Acc(a) \wedge \bot) \rightarrow Acc(c)$ (which is a tautology) is added to $\Sigma^0(G)$. The formulae $\neg Acc(b)$ and $\neg Acc(c)$ are deducible from $\Sigma_r^0(G)$.

Ex. 9 (cont'd) *a and e are unattacked: the formulae $Acc(a)$ and $Acc(e)$ are added to $\Sigma^0(G)$.*
c is attacked and defended: the formula $(Acc(a) \wedge Acc(e)) \rightarrow Acc(c)$ is added to $\Sigma^0(G)$. The formulae $Acc(c)$, $\neg Acc(b)$ and $\neg Acc(d)$ are deducible from $\Sigma_r^0(G)$.

Ex. 10 (cont'd) *a and d are unattacked: the formulae $Acc(a)$ and $Acc(d)$ are added to $\Sigma^0(G)$.*
c is attacked and defended: the formula $(Acc(a) \vee Acc(d)) \rightarrow Acc(c)$ is added to $\Sigma^0(G)$. The formulae $Acc(c)$ and $\neg Acc(b)$ are deducible from $\Sigma_r^0(G)$.

5.1.4 Stability

The idea is to propose a new property for expressing a kind of stability principle as it was done in [8]:

$\mathbf{P}_{\text{sta}}$ If an argument is not accepted, it must be attacked by an accepted argument.

And the associated formula:

$\mathbf{P}_{\text{sta}}$ $(\forall c)(\neg Acc(c) \rightarrow (\vee_{b \in \mathbf{R}^-(c)} Acc(b)))$

The stability principle can be explicitely encoded by adding to the base $\Sigma^0(G)$ (which describes the argumentation graph G) the formulas issued from $\mathbf{P}_{\text{sta}}$, one formula for each argument (either attacked or not). Let $\Sigma_s^0(G)$ denote the resulting base.
The formulae issued from $\mathbf{P}_{\text{acc}}^0$ can be written as follows:

- $(\forall c)((\vee_{b \in \mathbf{R}^-(c)} Acc(b)) \rightarrow NAcc(c))$
- $(\forall c)(NAcc(c) \rightarrow \neg Acc(c))$

Then, it is easy to prove that:

- $\mathbf{P}_{\text{def}}$ (resp. $\mathbf{P}_{\text{reins}}$) is a consequence of $\Sigma^0(G) \cup \mathbf{P}_{\text{sta}}$
- As an immediate consequence, Σ_s^0 classically entails Σ_d^0 and also Σ_r^0

Note that the above result is in line with the relation between stability and admissibility in the standard Dung's semantics.
Moreover, Ex. 4 on page 73 shows that the base $\Sigma_s^0(G)$ is not always logically consistent.

Let us illustrate $\Sigma_s^0(G)$ on different examples.

Ex. 1 (cont'd) *The formulae $Acc(a)$ and $\neg Acc(b) \rightarrow Acc(a)$ are added to $\Sigma^0(G)$.*

Ex. 3 (cont'd) *The formulae* $Acc(a)$, $\neg Acc(b) \rightarrow Acc(a)$ *and* $\neg Acc(c) \rightarrow Acc(b)$ *are added to* $\Sigma^0(G)$.

Ex. 6 (cont'd) *The formulae* $Acc(a)$, $Acc(d)$, $\neg Acc(b) \rightarrow Acc(a)$ *and* $\neg Acc(c) \rightarrow (Acc(b) \vee Acc(d))$ *are added to* $\Sigma^0(G)$.

Ex. 9 (cont'd) *The formulae* $Acc(a)$, $Acc(e)$, $\neg Acc(b) \rightarrow Acc(a)$, $\neg Acc(d) \rightarrow Acc(e)$ *and* $\neg Acc(c) \rightarrow (Acc(b) \vee Acc(d))$ *are added to* $\Sigma^0(G)$.

Ex. 10 (cont'd) *The formulae* $Acc(a)$, $Acc(d)$, $\neg Acc(b) \rightarrow (Acc(a) \vee Acc(d))$ *and* $\neg Acc(c) \rightarrow Acc(b)$ *are added to* $\Sigma^0(G)$.

5.1.5 Characterizing semantics for an AS in the basic language

In this section, we propose to characterize standard semantics using the principles previously defined and the corresponding bases, recalled in the following table:

Base		Used principles
$\Sigma^0(G)$	$=$	$\{\mathbf{P}^0_{\mathbf{acc}}\}$
$\Sigma^0_d(G)$	$= \Sigma^0(G) \cup$	$\{\mathbf{P}_{\mathbf{def}}\}$
$\Sigma^0_r(G)$	$= \Sigma^0(G) \cup$	$\{\mathbf{P}_{\mathbf{reins}}\}$
$\Sigma^0_s(G)$	$= \Sigma^0(G) \cup$	$\{\mathbf{P}_{\mathbf{sta}}\}$

Let AS = $\langle \mathbf{A}, \mathbf{R} \rangle$ and $S \subseteq \mathbf{A}$. We adopt the following notations:

- $Acc(S)$ denotes the set of literals $Acc(x)$ for $x \in S$,
- $\Phi_{Acc}(S)$ denotes the formula $\wedge_{x \in S} Acc(x)$.

Def. 21 *Let* $\mathcal{I}$ *be an interpretation of* Σ*, a set of formulae of the basic language,*

- $S_{\mathcal{I}} = \{x \in \mathbf{A} | \mathcal{I}(Acc(x)) = \textit{true}\}$
- $\mathcal{I}$ *is* $Acc(\mathbf{A})$*-maximal iff there is no interpretation* $\mathcal{I}'$ *of* Σ *such that* $S_{\mathcal{I}} \subset S_{\mathcal{I}'}$.
- $\mathcal{I}$ *is* $Acc(\mathbf{A})$*-minimal iff there is no interpretation* $\mathcal{I}'$ *of* Σ *such that* $S_{\mathcal{I}'} \subset S_{\mathcal{I}}$.

In other words, $\mathcal{I}$ is $Acc(\mathbf{A})$-maximal (resp. $Acc(\mathbf{A})$-minimal) if and only if the set of literals $Acc(x)$ satisfied by $\mathcal{I}$ is maximal (resp. minimal) for set-inclusion in $Acc(\mathbf{A})$.

Remark:
Let $\mathcal{I}$ be an interpretation of Σ, a set of formulae of the basic language, and $S \subseteq \mathbf{A}$. $\mathcal{I}$ is a model of $Acc(S)$ if and only if $S \subseteq S_{\mathcal{I}}$.

Characterizing conflict-free subsets

The models of $\Sigma^0(G)$ characterize the conflict-free subsets of G. Indeed, if $\mathcal{I}$ is a model of $\Sigma^0(G)$, $S_\mathcal{I}$ is conflict-free. The converse also holds as claimed in the following proposition:

Prop. 5 *S is conflict-free if and only if $\exists\mathcal{I}$ model of $\Sigma^0(G)$ such that $S_\mathcal{I} = S$.*

As an immediate consequence, we have a characterization of a conflict-free set S in terms of the set of literals $Acc(S)$:

Prop. 6 *S is conflict-free if and only if $Acc(S) \cup \Sigma^0(G)$ is consistent (or equivalently $\Sigma^0(G) \wedge \Phi_{Acc}(S)$ is consistent).*

Characterizing naive extensions

The models of $\Sigma^0(G)$ that maximize the satisfied literals of $Acc(\mathbf{A})$ characterize the naive extensions of G.

Prop. 7 *S is a naive extension if and only if $\exists\mathcal{I}$ $Acc(\mathbf{A})$-maximal model of $\Sigma^0(G)$ such that $S_\mathcal{I} = S$.*

We have also a characterization of a naive extension S in terms of the set of literals $Acc(S)$:

Prop. 8 *S is a naive extension if and only if $Acc(S)$ is a maximal (for set-inclusion) subset of $Acc(\mathbf{A})$ consistent with $\Sigma^0(G)$.*

Characterizing admissible subsets

The models of $\Sigma^0_d(G)$ characterize the admissible subsets of G. Indeed, if $\mathcal{I}$ is a model of $\Sigma^0_d(G)$, $S_\mathcal{I}$ is admissible. The converse also holds as claimed in the following proposition:

Prop. 9 *S is admissible if and only if $\exists\mathcal{I}$ model of $\Sigma^0_d(G)$ such that $S = S_\mathcal{I}$.*

As an immediate consequence of the above results, we have:

Prop. 10

1. *If S is admissible, then $Acc(S) \cup \Sigma^0_d(G)$ is consistent.*
2. *If $Acc(S) \cup \Sigma^0_d(G)$ is consistent, then there is an admissible set containing S.*

Note that we have only inclusion in the second part of the above proposition, as shown by Example 3 on page 71:

Ex. 3 (cont'd) *Let $S = \{c\}$. $Acc(S) \cup \Sigma^0_d(G)$ is consistent. However, S is not admissible whereas S is included in $\{a, c\}$ which is admissible.*

So we do not get an exact characterization of admissible sets S by testing the consistency of $Acc(S)$ with $\Sigma^0_d(G)$. Hopefully, an exact characterization holds for maximal (for set-inclusion) admissible sets.

Characterizing preferred extensions

The models of $\Sigma_d^0(G)$ that maximize the satisfied literals of $Acc(\mathbf{A})$ characterize the preferred extensions of G.

Prop. 11 *Let $S \subseteq A$. S is a preferred extension if and only if $\exists \mathcal{I}$ $Acc(\mathbf{A})$-maximal model of $\Sigma_d^0(G)$ such that $S_\mathcal{I} = S$.*

We have also a characterization of a preferred extension S in terms of $Acc(S)$:

Prop. 12 *S is a preferred extension if and only if $Acc(S)$ is a maximal (for set-inclusion) subset of $Acc(\mathbf{A})$ consistent with $\Sigma_d^0(G)$.*

Characterizing the grounded extension

It can be proved that there is only one model of $\Sigma_r^0(G)$ that minimizes the satisfied literals of $Acc(\mathbf{A})$. This model characterizes the grounded extension of G.

Prop. 13 *S is the grounded extension if and only if $S = S_\mathcal{I}$ where $\mathcal{I}$ is an $Acc(\mathbf{A})$-minimal model of $\Sigma_r^0(G)$.*

As a consequence we have:

Prop. 14 *The grounded extension is the set S_g defined by $S_g = \{x \in \mathbf{A} | \Sigma_r^0(G) \vdash Acc(x)\}$*

Characterizing the complete extensions

The models of $\Sigma_r^0(G) \cup \Sigma_d^0(G)$ characterize the complete extensions of G.

We recall that S is a complete extension if and only if S is admissible and each argument which is acceptable with respect to S belongs to S. In other words, S is a complete extension if and only if S is admissible and $\mathcal{F}(S) \subseteq S$.

We first characterize the subsets S such that $\mathcal{F}(S) \subseteq S$.

Prop. 15

1. *If $\mathcal{I}$ is a model of $\Sigma_r^0(G)$, then $\mathcal{F}(S_\mathcal{I}) \subseteq S_\mathcal{I}$.*
2. *If $\mathcal{F}(S) \subseteq S$ and S is conflict-free, then $\exists \mathcal{I}$ model of $\Sigma_r^0(G)$ such that $S = S_\mathcal{I}$.*

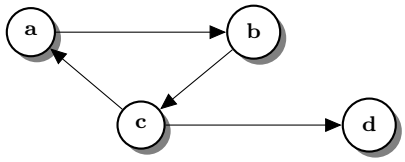
As a consequence we have the following characterization of complete extensions.

Prop. 16 *S is a complete extension if and only if $\exists \mathcal{I}$ model of $\Sigma_d^0(G) \cup \Sigma_r^0(G)$ such that $S = S_\mathcal{I}$.*

To sum up, the models of $\Sigma_d^0(G)$ enable to characterize the admissible sets (the conflict-free sets S such that $S \subseteq \mathcal{F}(S)$) whereas the models of $\Sigma_r^0(G)$ enable to characterize the conflict-free sets S such that $\mathcal{F}(S) \subseteq S$.

As a consequence, the models of $\Sigma_d^0(G)$ and $\Sigma_r^0(G)$ are generally different, as illustrated by the following examples:

Ex. 12



This example shows a model of $\Sigma_r^0(G)$ which is not a model of $\Sigma_d^0(G)$.

$\Sigma^0(G)$	$= \{$	$Acc(a) \rightarrow NAcc(b)$ $Acc(b) \rightarrow NAcc(c)$ $Acc(c) \rightarrow NAcc(a)$ $Acc(c) \rightarrow NAcc(d)$ $NAcc(a) \rightarrow \neg Acc(a)$ $NAcc(b) \rightarrow \neg Acc(b)$ $NAcc(c) \rightarrow \neg Acc(c)$ $NAcc(d) \rightarrow \neg Acc(d)$ $\}$
$\Sigma_d^0(G)$	$= \Sigma^0(G) \cup \{$	$Acc(a) \rightarrow Acc(b)$ $Acc(b) \rightarrow Acc(c)$ $Acc(c) \rightarrow Acc(a)$ $Acc(d) \rightarrow Acc(b)$ $\}$
$\Sigma_r^0(G)$	$= \Sigma^0(G) \cup \{$	$Acc(a) \rightarrow Acc(c)$ $Acc(b) \rightarrow Acc(a)$ $Acc(c) \rightarrow Acc(b)$ $Acc(b) \rightarrow Acc(d)$ $\}$

Let $\mathcal{I}$ be defined by: $\forall x \in \mathbf{A}, x \neq d$, $\mathcal{I}(Acc(x)) =$ false, $\mathcal{I}(Acc(d)) =$ true and $\forall x \in \mathbf{A}, \mathcal{I}(NAcc(x)) =$ false if and only if $\mathcal{I}(Acc(x)) =$ true.
$\mathcal{I}$ is a model of $\Sigma_r^0(G)$ and not a model of $\Sigma_d^0(G)$.

Example 3 on page 71 shows a model of $\Sigma_d^0(G)$ which is not a model of $\Sigma_r^0(G)$.

Ex. 3 (cont'd) *Let $\mathcal{I}$ be defined by: $\mathcal{I}(Acc(a)) =$ true, $\mathcal{I}(Acc(b)) = \mathcal{I}(Acc(c)) =$ false and $\forall x \in \mathbf{A}, \mathcal{I}(NAcc(x)) =$ false if and only if $\mathcal{I}(Acc(x)) =$ true. $\mathcal{I}$ is a model of $\Sigma_d^0(G)$ and not a model of $\Sigma_r^0(G)$.*

And finally, we have a characterization of a complete extension S in terms of the set of literals $Acc(S)$:

Prop. 17 *S is a complete extension if and only if $Acc(S) \cup \{\neg Acc(x) | x \in \mathbf{A} \setminus S\} \cup \Sigma_d^0(G) \cup \Sigma_r^0(G)$ is consistent*
if and only if $Acc(S) \cup \{NAcc(x) | x \in \mathbf{A} \setminus S\} \cup \Sigma_d^0(G) \cup \Sigma_r^0(G)$ is consistent.

Characterizing the stable extensions

The models of $\Sigma_s^0(G)$ characterize the stable subsets of G. Indeed, if $\mathcal{I}$ is a model of $\Sigma_s^0(G)$, $S_\mathcal{I}$ is stable. The converse also holds as claimed in the following proposition:

Prop. 18 *S is a stable extension if and only if $\exists \mathcal{I}$ model of $\Sigma_s^0(G)$ such that $S_\mathcal{I} = S$.*

Stable extensions can be also characterized in terms of the set of literals $Acc(S)$:

Prop. 19 *S is a stable extension if and only if $Acc(S) \cup \{\neg Acc(x) | x \in \mathbf{A} \setminus S\} \cup \Sigma_s^0(G)$ is consistent if and only if $Acc(S) \cup \{NAcc(x) | x \in \mathbf{A} \setminus S\} \cup \Sigma_s^0(G)$ is consistent.*

Other characterizations can be given, in terms of the set of literals $Acc(S)$ and in terms of models:

Prop. 20 *S is a stable extension if and only if:*

1. *$Acc(S)$ is a maximal for set-inclusion subset of $Acc(\mathbf{A})$ consistent with $\Sigma^0(G)$, and*
2. *$\forall a \notin S, (\Sigma^0(G) \cup Acc(S)) \vdash NAcc(a)$*

Prop. 21 *S is a stable extension if and only if:*

1. *$\exists \mathcal{I}$ model of $\Sigma^0(G)$ such that $S = S_{\mathcal{I}}$, and*
2. *$\forall \mathcal{J}$ model of $\Sigma^0(G)$ such that $S \subseteq S_{\mathcal{J}}$, $\forall a \notin S, \mathcal{J}(NAcc(a)) =$ true.*

The following table synthetizes the links between properties of an interpretation $\mathcal{I}$ and properties of its associated set $S_{\mathcal{I}} = \{x \in \mathbf{A} | \mathcal{I}(Acc(x)) =$ *true*$\}$.

$\mathcal{I}$	$S_{\mathcal{I}}$	Number of Prop.
model of $\Sigma^0(G)$	conflict-free	5, 6
$Acc(\mathbf{A})$-maximal model of $\Sigma^0(G)$	naive extension	7, 8
model of $\Sigma_d^0(G)$	admissible	9, 10
$Acc(\mathbf{A})$-maximal model of $\Sigma_d^0(G)$	preferred extension	11, 12
$Acc(\mathbf{A})$-minimal model of $\Sigma_r^0(G)$	grounded extension	13, 14
model of $\Sigma_d^0(G) \cup \Sigma_r^0(G)$	complete extension	16, 17
model of $\Sigma_s^0(G)$	stable extension	18, 19

5.1.6 Basic principles restated in terms of $NAcc$

The principles described above (the conflict-free, defence and reinstatement principles) are expressed in terms of acceptability of interacting arguments. The defence principle for an argument c for instance is expressed by a formula that involves the literal $Acc(c)$ and the literals $Acc(a)$ such that a attacks an attacker of c.

In this section, we consider new principles expressed under the form of relations between $NAcc(x)$ and $Acc(y)$ when x and y interact. We show that these new principles can be considered as building blocks from which the defence and reinstatement principles can be defined.

The new principles

Let us recall that the intended meaning of $NAcc(x)$ is "x cannot be accepted". The two following principles give conditions for $NAcc(c)$ being true in terms of acceptability of the attackers of c.

P5 If an argument cannot be accepted, then at least one of its attackers must be accepted.

P6 If an argument is attacked by at least one accepted argument, then it cannot be accepted.

More formally, we obtain the following formulas:

P5 $(\forall c)(NAcc(c) \rightarrow (\vee_{b \in \mathbf{R}^-(c)} Acc(b)))$

P6 $(\forall c)((\vee_{b \in \mathbf{R}^-(c)} Acc(b)) \rightarrow NAcc(c))$

Obvioulsy, the above formula **P6** is a direct consequence of Property $\mathbf{P}^0_{\mathbf{acc}}$. So it is already entailed by the base $\Sigma^0(G)$.

The next two principles give conditions for $Acc(c)$ being true in terms of literals $NAcc(b)$ where b is an attacker of c.

P7 An argument may be accepted only if each of its attackers cannot be accepted.

P8 An argument is accepted provided that each of its attackers cannot be accepted.

More formally, we obtain the following formulas:

P7 $(\forall c)(Acc(c) \rightarrow (\wedge_{b \in \mathbf{R}^-(c)} NAcc(b)))$

P8 $(\forall c)((\wedge_{b \in \mathbf{R}^-(c)} NAcc(b)) \rightarrow Acc(c))$

The new principles wrt $\mathbf{P}_{\mathbf{def}}$, $\mathbf{P}_{\mathbf{reins}}$ and $\mathbf{P}_{\mathbf{sta}}$
The following results are easy to prove:

1. **P7** is a consequence of the conjunction of **P6** and $\mathbf{P}_{\mathbf{def}}$
2. $\mathbf{P}_{\mathbf{def}}$ is a consequence of the conjunction of **P5** and **P7**
3. $\mathbf{P}_{\mathbf{reins}}$ is a consequence of the conjunction of **P6** and**P8**
4. **P8** is a consequence of the conjunction of **P5** and $\mathbf{P}_{\mathbf{reins}}$

In other words,

- From **P6**, we have that $\mathbf{P}_{\mathbf{def}}$ entails **P7**
- From **P5**, we have that **P7** entails $\mathbf{P}_{\mathbf{def}}$

and

- From **P6**, we have that **P8** entails $\mathbf{P}_{\mathbf{reins}}$
- From **P5**, we have that $\mathbf{P}_{\mathbf{reins}}$ entails **P8**

Moreover, using the formulae $NAcc(x) \rightarrow \neg Acc(x)$ it is easy to prove that:

- **P8** (resp. **P5**) is a consequence of $\mathbf{P}_{\mathbf{sta}}$

Characterizing the new principles in terms of models of $\Sigma^0(G)$

Adding Property **P5** to $\Sigma^0(G)$ comes to consider the completion of the first kind of rules issued from Property $\mathbf{P}^0_{\mathbf{acc}}$ wrt the predicate $NAcc$. That amounts to restrict the models of $\Sigma^0(G)$ to those models where the set of literals $NAcc(x)$ satisfied is in some sense minimal (for set-inclusion).

Let $\mathcal{I}$ be an interpretation of Σ, a set of formulae of the basic language. We recall that $S_{\mathcal{I}} = \{x \in \mathbf{A} | \mathcal{I}(Acc(x)) = \textit{true}\}$.
Let us define :

- $N_{\mathcal{I}} = \{x \in \mathbf{A} | \mathcal{I}(NAcc(x)) = \textit{true}\}$
- $\mathcal{I}$ is $NAcc(\mathbf{A})$-minimal iff there is no interpretation $\mathcal{I}'$ of Σ such that $S_{\mathcal{I}'} = S_{\mathcal{I}}$ and $N_{\mathcal{I}'} \subset N_{\mathcal{I}}$.

We have the following result :

Prop. 22 *The models of $\Sigma^0(G) \cup \{\mathbf{P5}\}$ are exactly the $NAcc(\mathbf{A})$-minimal models of $\Sigma^0(G)$.*

5.2 Semantics for an AS in the extended language

In this section, we characterize standard semantics of an AS by logical formulas of the extended language introduced in Section 4.2 on page 23. Note that the argumentation graph G associated with an AS is now represented by the base $\Sigma(G)$.

As noted in Section 4.2.2 on page 24, in presence of attacks which are not attacked, formulae issued from $\mathbf{P}_{\mathbf{val}}$ are tautologies. So there is no sufficient condition for an attack being valid. As a consequence it is not assumed that non attacked attacks are always valid. This property will appear later in Section 6.1.3 on page 53 as a consequence of a kind of reinstatement principle for attacks.
However, in the particular case of an AS (with simple attacks), standard Dung's semantics handle attacks as if they were valid. So in order to recover standard Dung's semantics in the case of an AS, we will have to assume that each attack is valid. In Section 5.2.1, we will consider a specific principle for AS and its logical expression in the extended language.

Then, as in Section 5.1 on page 33, we will consider the basic principles (conflict-freeness, defence, reinstatement and stability) and give a logical expression for each of them in the extended language.

5.2.1 Specificity of an AS wrt validity

Standard semantics of an AS can be captured in the extended language, due to the addition of the following property:

$\mathbf{P}^{\mathbf{as}}_{\mathbf{val}}$ **:** Each attack is valid.

Property $\mathbf{P}^{\mathbf{as}}_{\mathbf{val}}$ can be expressed by the following formula:

$\mathbf{P}^{\mathbf{as}}_{\mathbf{val}}$ **:** $(\forall x \in Attack)(Val(x))$

This formula does not belong to $\Sigma(G)$. Adding this formula to the base $\Sigma(G)$ leads to a base denoted by $\Sigma_{as}(G)$.

Due to $\mathbf{P}^{\mathbf{as}}_{\mathbf{val}}$, $\Sigma_{as}(G)$ entails the formulas $Act(\alpha) \leftrightarrow Gr(\alpha) \leftrightarrow Acc(s_\alpha)$ for each attack α. And the formula $(Val(\alpha) \wedge Acc(s_\alpha)) \rightarrow NAcc(t_\alpha)$ is equivalent to the formula $Acc(s_\alpha) \rightarrow NAcc(t_\alpha)$.
The above remark enables us to consider a simplified form of $\Sigma_{as}(G)$ that only contains literals of the form $a = s_\alpha$, $b = t_\alpha$ and complex formulas of the type $Acc(s_\alpha) \rightarrow NAcc(t_\alpha)$ and $NAcc(t_\alpha) \rightarrow \neg Acc(t_\alpha)$.
Indeed, we obtain a base of formulas which is very close to the base $\Sigma^0(G)$ used in Section 5.1 on page 33.

Note also that each model $\mathcal{I}$ of the simplified form of $\Sigma_{as}(G)$ can be extended to a model of the initial base $\Sigma_{as}(G)$ by taking $\forall \alpha, \mathcal{I}(Val(\alpha)) = \textit{true}$, $\mathcal{I}(Gr(\alpha)) = \mathcal{I}(Act(\alpha)) = \mathcal{I}(Acc(s_\alpha))$. For simplicity, the simplified form of $\Sigma_{as}(G)$ will still be called $\Sigma_{as}(G)$.

In the rest of Section 5.2 on the preceding page, the base corresponding to the translation of an AS with the extended language will be $\Sigma_{as}(G)$.

5.2.2 Conflict-freeness

The interaction α corresponding to the attack $a\mathbf{R}b$ is encoded using the symbols a, b, α, s_α, t_α, the literals $a = s_\alpha$, $b = t_\alpha$ and the formulas $Acc(s_\alpha) \rightarrow NAcc(t_\alpha))$ and $NAcc(t_\alpha) \rightarrow \neg Acc(t_\alpha)$.
So the base $\Sigma_{as}(G)$ already contains formulas that express the fact that two conflicting arguments cannot be jointly accepted.

5.2.3 Defence

As in Section 5.1.2 on page 34, the defence principle can be explicitely encoded by adding to the base $\Sigma_{as}(G)$ the formulas issued from Property $\mathbf{P}_{\mathbf{def}}$, one formula for each attack. However, those formulae must be written in the extended language.

$\mathbf{P}_{\mathbf{def}}$ $(\forall \alpha \in Attack)(Acc(t_\alpha) \rightarrow (\exists \beta \in Attack)(t_\beta = s_\alpha \wedge Acc(s_\beta)))$

Remark: In Section 5.1.2 on page 34, $\mathbf{P}_{\mathbf{def}}$ is expressed for each attacked argument c and the resulting formula is equivalent to a conjunction of formulas, one for each attack to c. In contrast, the extended language enables us to work directly at the level of attacks. So applying $\mathbf{P}_{\mathbf{def}}$ produces one formula for each attack α.
Note also that there is no need for considering that α or β should be active or valid, as we consider the particular case of an AS and a simplified form of $\Sigma_{as}(G)$ as explained above.

Let $\Sigma_d(G)$ still denote the resulting base. Let us illustrate $\Sigma_d(G)$ on different examples (always in the simplified form as it is described for Example 1 on page 67 in Section 4.2 on page 23). Note that we recover the same results as those obtained in Section 5.1.2 on page 34.

Ex. 1 (cont'd) *The formula* $\neg Acc(b)$ *is added to* $\Sigma_{as}(G)$

Ex. 3 (cont'd) *The formulae* $\neg Acc(b)$ *and* $Acc(c) \rightarrow Acc(a)$ *are added to* $\Sigma_{as}(G)$

Ex. 6 (cont'd) *The formulae* $\neg Acc(b)$ *and* $\neg Acc(c)$ *are added to* $\Sigma_{as}(G)$

Ex. 9 (cont'd) *The formulae* $\neg Acc(d)$, $\neg Acc(b)$, $Acc(c) \rightarrow Acc(a)$ *and* $Acc(c) \rightarrow Acc(e)$ *(so* $Acc(c) \rightarrow (Acc(a) \wedge Acc(e))$*) are added to* $\Sigma_{as}(G)$

Ex. 10 (cont'd) *The formulae* $\neg Acc(b)$ *and* $Acc(c) \rightarrow (Acc(a) \vee Acc(d))$ *are added to* $\Sigma_{as}(G)$

Note that, as in Section 5.1.2 on page 34, the base $\Sigma_d(G)$ is consistent:

Prop. 23 *Let* G *be an argumentation graph representing an AS with* $\mathbf{A}$ *the set of arguments and* $\mathbf{R} \subseteq \mathbf{A} \times \mathbf{A}$ *the set of attacks. The logical knowledge base* $\Sigma_d(G)$ *is consistent.*

5.2.4 Reinstatement

As in Section 5.1.3 on page 35, the reinstatement principle can be explicitely encoded by adding to the base $\Sigma_{as}(G)$ the formulas issued from the property $\mathbf{P}_{\mathbf{reins}}$, one formula for each argument. However, those formulae must be written in the extended language.

$\mathbf{P}_{\mathbf{reins}}$ $(\forall c \in Argument)(((\forall \alpha \in Attack)(t_\alpha = c \rightarrow (\exists \beta \in Attack)(t_\beta = s_\alpha \wedge Acc(s_\beta)))) \rightarrow Acc(c))$

Remark: As a logical consequence of $\mathbf{P}_{\mathbf{reins}}$, we obtain the formula:

$$(\forall c \in Argument)(((\forall \alpha \in Attack)(\neg(t_\alpha = c))) \rightarrow Acc(c))$$

which is equivalent to:

$$(\forall c \in Argument)(\neg((\exists \alpha \in Attack)(t_\alpha = c)) \rightarrow Acc(c))$$

So in the case of an unattacked argument c, as $\neg((\exists \alpha \in Attack)(t_\alpha = c))$ is *true*, we can deduce $Acc(c)$.

Let $\Sigma_r(G)$ still denote the resulting base. Let us illustrate $\Sigma_r(G)$ on different examples. Note that we recover the same results as those obtained in Section 5.1.3 on page 35.

Ex. 1 (cont'd) a *is not attacked: the formula* $Acc(a)$ *is added to* $\Sigma_r(G)$*. Then,* $\neg Acc(b)$ *is deducible from* $\Sigma_r(G)$.

Ex. 3 (cont'd) a *is not attacked: the formula* $Acc(a)$ *is added to* $\Sigma_r(G)$.
c *is attacked and defended: the formula* $Acc(a) \rightarrow Acc(c)$ *is added to* $\Sigma(G)$*. The formulae* $Acc(c)$ *and* $\neg Acc(b)$ *are deducible from* $\Sigma_r(G)$.

Ex. 6 (cont'd) a *and* d *are unattacked: the formulae* $Acc(a)$ *and* $Acc(d)$ *are added to* $\Sigma_{as}(G)$.
c *is attacked with two attackers: the formula* $(Acc(a) \wedge \bot) \rightarrow Acc(c)$ *(a tautology) is added to* $\Sigma_{as}(G)$*. The formulae* $\neg Acc(c)$ *and* $\neg Acc(b)$ *are deducible from* $\Sigma_r(G)$.

Ex. 9 (cont'd) a *and* e *are unattacked: the formulae* $Acc(a)$ *and* $Acc(e)$ *are added to* $\Sigma_{as}(G)$.
c *is attacked and defended: the formula* $(Acc(a) \wedge Acc(e)) \rightarrow Acc(c)$ *is added to* $\Sigma_{as}(G)$*. The formulae* $Acc(c)$, $\neg Acc(b)$ *and* $\neg Acc(d)$ *are deducible from* $\Sigma_r(G)$.

Ex. 10 (cont'd) *a and d are unattacked: the formulae $Acc(a)$ and $Acc(d)$ are added to $\Sigma_{as}(G)$. c is attacked and defended: the formula $(Acc(a) \vee Acc(d)) \rightarrow Acc(c)$ is added to $\Sigma_{as}(G)$. The formulae $Acc(c)$ and $\neg Acc(b)$ are deducible from $\Sigma_r(G)$.*

Note that, as in Section 5.1.3 on page 35, the base $\Sigma_r(G)$ is consistent:

Prop. 24 *Let G be an argumentation graph representing an AS with $\mathbf{A}$ the set of arguments and $\mathbf{R} \subseteq \mathbf{A} \times \mathbf{A}$ the set of attacks. The logical knowledge base $\Sigma_r(G)$ is consistent.*

5.2.5 Stability

The principle $\mathbf{P}_{\mathbf{sta}}$ can be expressed using the extended language as follows:

$\mathbf{P}_{\mathbf{sta}}$ $(\forall c \in Argument)(\neg Acc(c) \rightarrow (\exists \beta \in Attack)((t_\beta = c) \wedge Acc(s_\beta)))$

This stability principle can be explicitely encoded by adding to the base $\Sigma_{as}(G)$ (which describes the argumentation graph G corresponding to an AS) the formulas issued from $\mathbf{P}_{\mathbf{sta}}$, one formula for each argument (either attacked or not). Let $\Sigma_s(G)$ denote the resulting base.

Note that, as in Section 5.1.4 on page 37, the base $\Sigma_s(G)$ can be inconsistent (once again consider Ex. 4 on page 73).

Let us illustrate $\Sigma_s(G)$ on different examples.

Ex. 1 (cont'd) *The formulae $Acc(a)$ and $\neg Acc(b) \rightarrow Acc(a)$ are added to $\Sigma_{as}(G)$.*

Ex. 3 (cont'd) *The formulae $Acc(a)$, $\neg Acc(b) \rightarrow Acc(a)$ and $\neg Acc(c) \rightarrow Acc(b)$ are added to $\Sigma_{as}(G)$.*

Ex. 6 (cont'd) *The formulae $Acc(a)$, $Acc(d)$, $\neg Acc(b) \rightarrow Acc(a)$ and $\neg Acc(c) \rightarrow (Acc(b) \vee Acc(d))$ are added to $\Sigma_{as}(G)$.*

Ex. 9 (cont'd) *The formulae $Acc(a)$, $Acc(e)$, $\neg Acc(b) \rightarrow Acc(a)$, $\neg Acc(d) \rightarrow Acc(e)$ and $\neg Acc(c) \rightarrow (Acc(b) \vee Acc(d))$ are added to $\Sigma_{as}(G)$.*

Ex. 10 (cont'd) *The formulae $Acc(a)$, $Acc(d)$, $\neg Acc(b) \rightarrow (Acc(a) \vee Acc(d))$ and $\neg Acc(c) \rightarrow Acc(b)$ are added to $\Sigma_{as}(G)$.*

5.2.6 Characterizing semantics for an AS in the extended language

In this section, we propose to characterize standard semantics using the principles previously defined and the corresponding bases, recalled in the following table:

Base		Used principles
$\Sigma(G)$	$=$	$\{\mathbf{P}_{\mathbf{acc}}, \mathbf{P}_{\mathbf{val}}, \mathbf{P}_{\mathbf{gr}}, \mathbf{P}_{\mathbf{act}}\}$
$\Sigma_{as}(G)$	$= \Sigma(G) \cup$	$\{\mathbf{P}^{\mathbf{as}}_{\mathbf{val}}\}$
$\Sigma_d(G)$	$= \Sigma_{as}(G) \cup$	$\{\mathbf{P}_{\mathbf{def}}\}$
$\Sigma_r(G)$	$= \Sigma_{as}(G) \cup$	$\{\mathbf{P}_{\mathbf{reins}}\}$
$\Sigma_s(G)$	$= \Sigma_{as}(G) \cup$	$\{\mathbf{P}_{\mathbf{sta}}\}$

Let us first recall that we work with the simplified form of the base $\Sigma_{as}(G)$. That implies the use of the constant symbols α, t_α, s_α and only the predicate symbols Acc and $NAcc$, as in $\Sigma^0(G)$. So, we may still consider the set of literals $S_\mathcal{I}$ for $\mathcal{I}$ being an interpretation of $\Sigma_{as}(G)$.

For instance, $S \subseteq A$ being conflict-free can be expressed by the following formula:

$$(\forall a \in S)(\forall b \in S)(\forall \alpha)(\neg((a = s_\alpha) \wedge (b = t_\alpha)))$$

or equivalently[1]:

$$(\forall \alpha)(\neg(s_\alpha \in S \wedge t_\alpha \in S))$$

All the characterizations obtained in Section 5.1.5 on page 38 can be extended in a straightforward way, replacing $\Sigma^0(G)$ by $\Sigma_{as}(G)$. The proofs can be adapted using t_α and s_α for denoting arguments.

Prop. 25

1. *S is conflict-free if and only if $\exists \mathcal{I}$ model of $\Sigma_{as}(G)$ such that $S_\mathcal{I} = S$.*
2. *S is a naive extension if and only if $\exists \mathcal{I}$ $Acc(\mathbf{A})$-maximal model of $\Sigma_{as}(G)$ such that $S_\mathcal{I} = S$.*
3. *S is admissible if and only if $\exists \mathcal{I}$ model of $\Sigma_d(G)$ such that $S = S_\mathcal{I}$.*
4. *S is a preferred extension if and only if $\exists \mathcal{I}$ $Acc(\mathbf{A})$-maximal model of $\Sigma_d(G)$ such that $S_\mathcal{I} = S$.*
5. *S is the grounded extension if and only if $S = S_\mathcal{I}$ where $\mathcal{I}$ is the $Acc(\mathbf{A})$-minimal model of $\Sigma_r(G)$.*
6. *S is a complete extension if and only if $\exists \mathcal{I}$ model of $\Sigma_d(G) \cup \Sigma_r(G)$ such that $S = S_\mathcal{I}$.*
7. *S is a stable extension if and only if $\exists \mathcal{I}$ model of $\Sigma_s(G)$ such that $S_\mathcal{I} = S$.*
8. *S is a stable extension if and only if $\exists \mathcal{I}$ model of $\Sigma_{as}(G)$ such that $S = S_\mathcal{I}$, and $\forall \mathcal{J}$ model of $\Sigma_{as}(G)$ such that $S \subseteq S_\mathcal{J}$, $\forall a \notin S, \mathcal{J}(NAcc(a)) =$ true.*

and in terms of $Acc(S)$:

Prop. 26

1. *S is conflict-free if and only if $Acc(S) \cup \Sigma_{as}(G)$ is consistent.*
2. *S is a naive extension if and only if $Acc(S)$ is a maximal (for set-inclusion) subset of $Acc(\mathbf{A})$ consistent with $\Sigma_{as}(G)$.*
3. *If S is admissible, then $Acc(S) \cup \Sigma_d(G)$ is consistent. If $Acc(S) \cup \Sigma_d(G)$ is consistent, then there is an admissible set containing S.*
4. *S is a preferred extension if and only if $Acc(S)$ is a maximal (for set-inclusion) subset of $Acc(\mathbf{A})$ consistent with $\Sigma_d(G)$.*

[1] $s_\alpha \in S$ is equivalent to $(\exists a \in S)(a = s_\alpha)$.

5. *The grounded extension is the set S_g defined by $S_g = \{x \in \mathbf{A} | \Sigma_r(G) \vdash Acc(x)\}$.*

6. *S is a complete extension if and only if $Acc(S) \cup \{\neg Acc(x) | x \in \mathbf{A} \setminus S\} \cup \Sigma_d(G) \cup \Sigma_r(G)$ is consistent.*
   S is a complete extension if and only if $Acc(S) \cup \{NAcc(x) | x \in \mathbf{A} \setminus S\} \cup \Sigma_d(G) \cup \Sigma_r(G)$ is consistent.

7. *S is a stable extension if and only if $Acc(S) \cup \{\neg Acc(x) | x \in \mathbf{A} \setminus S\} \cup \Sigma_s(G)$ is consistent*
   S is a stable extension if and only if $Acc(S) \cup \{NAcc(x) | x \in \mathbf{A} \setminus S\} \cup \Sigma_s(G)$ is consistent.

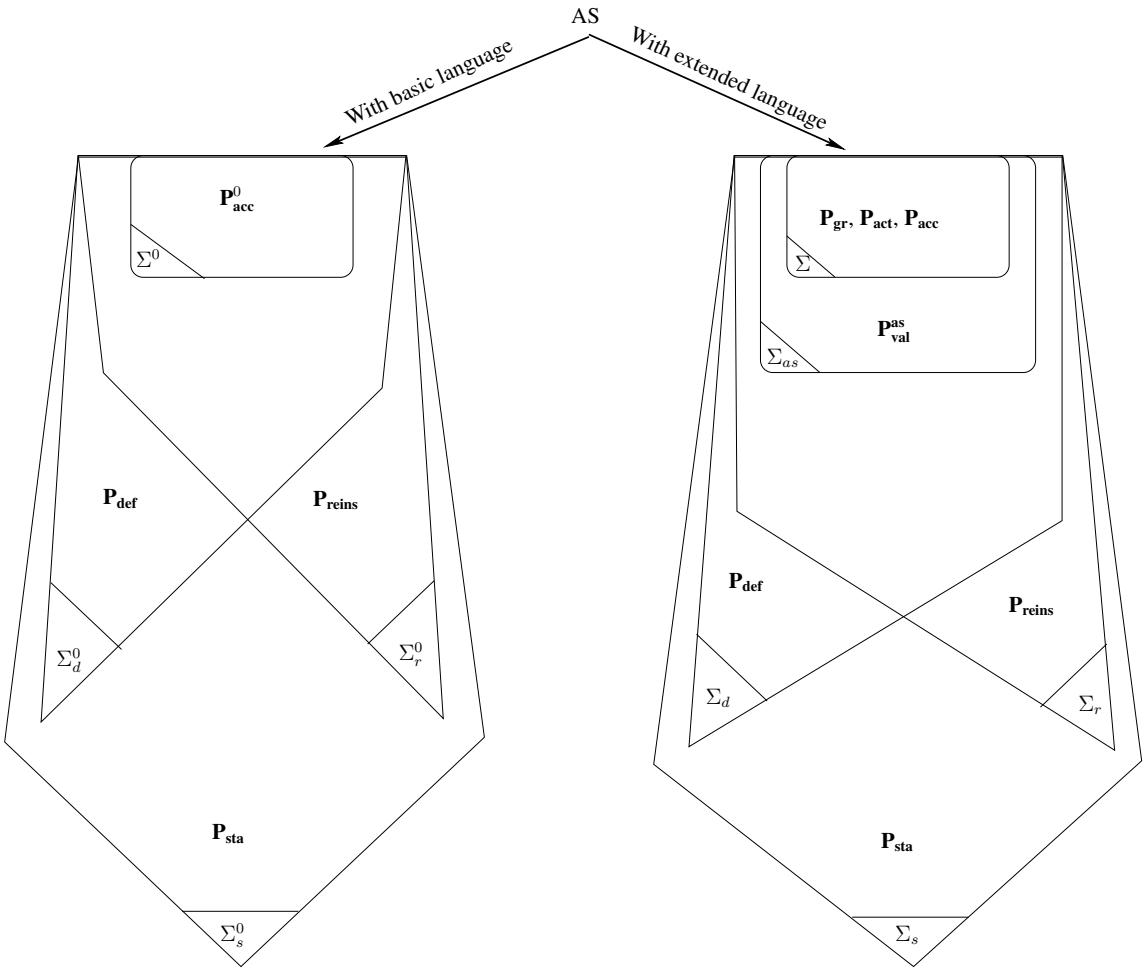
8. *S is a stable extension if and only if $Acc(S)$ is a maximal for set-inclusion subset of $Acc(\mathbf{A})$ consistent with $\Sigma_{as}(G)$, and $\forall a \notin S, (\Sigma_{as}(G) \cup Acc(S)) \vdash NAcc(a)$.*

The following table synthetizes all these results in both cases (basic language or extended language) giving the links between properties of an interpretation $\mathcal{I}$ and properties of its associated set $S_{\mathcal{I}} = \{x \in \mathbf{A} | \mathcal{I}(Acc(x)) = \textit{true}\}$.

$\mathcal{I}$	$S_{\mathcal{I}}$	Number of Prop.	
		basic lg.	extended lg.
model of $\Sigma^0(G)$ (resp. $\Sigma_{as}(G)$)	conflict-free	5, 6	25.1, 26.1
$Acc(\mathbf{A})$-maximal model of $\Sigma^0(G)$ (resp. $\Sigma_{as}(G)$)	naive extension	7, 8	25.2, 26.2
model of $\Sigma^0_d(G)$ (resp. $\Sigma_d(G)$)	admissible	9, 10	25.3, 26.3
$Acc(\mathbf{A})$-maximal model $\Sigma^0_d(G)$ (resp. $\Sigma_d(G)$)	preferred extension	11, 12	25.4, 26.4
$Acc(\mathbf{A})$-minimal model of $\Sigma^0_r(G)$ (resp. $\Sigma_r(G)$)	grounded extension	13, 14	25.5, 26.5
model of $\Sigma^0_d(G) \cup \Sigma^0_r(G)$ (resp. $\Sigma_d(G) \cup \Sigma_r(G)$)	complete extension	16, 17	25.6, 26.6
model of $\Sigma^0_s(G)$ (resp. $\Sigma_s(G)$)	stable extension	18, 19	25.7, 26.7

5.3 Synthesis for AS

The following figure describes the different encodings that can be used for an AS.


AS
With basic language
With extended language
$\mathbf{P}^0_{\mathbf{acc}}$
Σ^0
$\mathbf{P}_{\mathbf{def}}$
$\mathbf{P}_{\mathbf{reins}}$
Σ^0_d
Σ^0_r
$\mathbf{P}_{\mathbf{sta}}$
Σ^0_s
$\mathbf{P}_{\mathbf{gr}}$, $\mathbf{P}_{\mathbf{act}}$, $\mathbf{P}_{\mathbf{acc}}$
Σ
$\mathbf{P}^{\mathbf{as}}_{\mathbf{val}}$
Σ_{as}
$\mathbf{P}_{\mathbf{def}}$
$\mathbf{P}_{\mathbf{reins}}$
Σ_d
Σ_r
$\mathbf{P}_{\mathbf{sta}}$
Σ_s

Chapter 6

Semantics for an ASAF

We propose to :

- consider again the basic principles (conflict-freeness, defence, reinstatement, stability) taking into account the fact that attack could be attacked,
- give a logical expression for each of these principles in the extended language, thus leading to add formulas to the base $\Sigma(G)$ and producing new bases in the same way as $\Sigma_d(G)$, $\Sigma_r(G)$ and $\Sigma_s(G)$ were produced,
- study examples of such bases encoding argumentation graphs with recursive attacks,
- define standard semantics for an ASAF,
- provide a characterization of the associated extensions in terms of models of the new bases $\Sigma(G)$, $\Sigma_d(G)$, $\Sigma_r(G)$, and $\Sigma_s(G)$.

6.1 Basic principles revisited with recursive attacks

In presence of recursive interactions, the conflict-freeness, defence, reinstatement and stability principles must be reformulated for taking into account the fact that attacks could be not valid. Indeed, the fact that an attack α is grounded is taken into account through the literal $Acc(s_\alpha)$, due to Property $\mathbf{P_{gr}}$. Moreover, due to Property $\mathbf{P_{act}}$, we could do without the predicate symbol Act. So we concentrate on the notion of validity.

The first idea is to extend the definition of S being conflict-free by allowing non-valid attacks between arguments in S.
Similarly, a could be defended by S provided that S weakens each attack α to a, either by attacking s_α, or by attacking α itself.

6.1.1 Conflict-freeness

In presence of recursive attacks, the conflict-free property can be reformulated as follows:

$\mathbf{P}^{\mathbf{arg}}_{\mathbf{cf}}$ If there is a valid attack between two arguments, they cannot be jointly accepted.

Note that the fact that two arguments may be conflicting depends on the validity of the attack between them.

The attack $\alpha = (a, b)$ is encoded in the base $\Sigma(G)$ by the formulae $(Val(\alpha) \wedge Acc(a)) \rightarrow NAcc(b)$ and $NAcc(b) \rightarrow \neg Acc(b)$. So the base $\Sigma(G)$ already contains formulas that express the property $\mathbf{P}^{\mathbf{arg}}_{\mathbf{cf}}$.

Similarly we could consider a kind of conflict-freeness principle for attacks.

$\mathbf{P}^{\mathbf{att}}_{\mathbf{cf}}$ If there is an attack from an accepted argument to an attack, these attacks cannot be both valid.

Due to the formulae encoding an attack $\alpha = (a, \beta)$ in the base $\Sigma(G)$, this property is already expressed in $\Sigma(G)$ (see $\mathbf{P}_{\mathbf{val}}$ and its logical translation).

6.1.2 Defence

Similarly, a could be defended by S provided that S weakens each valid attack α to a, either by attacking s_α, or by attacking α itself.

In other words, the idea is to claim that an argument a is acceptable wrt a set of arguments S (in other words a is defended by S) if S weakens each attack α to a, either by attacking s_α, or by attacking α itself. Moreover, the defense should be obtained with valid attacks.

So, we propose to reformulate the defence principle as follows:

$\mathbf{P}^{\mathbf{arg}}_{\mathbf{def}}$**:** An attacked argument may be accepted only if for each attack against it, either the source or the attack itself is in turn attacked by a valid attack from an accepted argument (*i.e.* an active attack).

The property $\mathbf{P}^{\mathbf{arg}}_{\mathbf{def}}$ is expressed by the following formula:

$\mathbf{P}^{\mathbf{arg}}_{\mathbf{def}}$ $(\forall \alpha \in Attack)(Acc(t_\alpha) \rightarrow (\exists \beta \in Attack)(t_\beta \in \{s_\alpha, \alpha\} \wedge Val(\beta) \wedge Acc(s_\beta)))$

Similarly we consider a kind of defence principle for attacks.

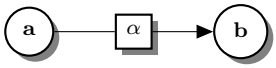
$\mathbf{P}^{\mathbf{att}}_{\mathbf{def}}$ An attack may be valid only if for each attack against it, either the source or the attack itself is in turn attacked by a valid attack from an accepted argument.

The property $\mathbf{P}^{\mathbf{att}}_{\mathbf{def}}$ is expressed by the following formula:

$\mathbf{P}^{\mathbf{att}}_{\mathbf{def}}$ $(\forall \alpha \in Attack)(\forall \delta \in Attack)(((\delta = t_\alpha) \wedge Val(\delta)) \rightarrow (\exists \beta \in Attack)(t_\beta \in \{s_\alpha, \alpha\} \wedge Val(\beta) \wedge Acc(s_\beta)))$

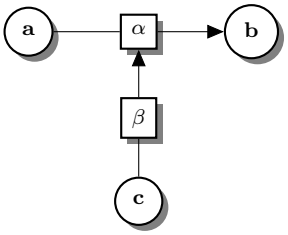
The formulae that are obtained from $\mathbf{P}^{\mathbf{arg}}_{\mathbf{def}}$ and $\mathbf{P}^{\mathbf{att}}_{\mathbf{def}}$ are added to the base $\Sigma(G)$, thus producing the base $\Sigma_d(G)$.

Two particular cases: Consider the following graph:



Adding $\mathbf{P}^{\mathbf{arg}}_{\mathbf{def}}$ to Σ_G enables to produce the formula $\neg Acc(b)$ since there is no attack from α or its source.

Consider now the following graph:



Adding $\mathbf{P}^{\mathbf{att}}_{\mathbf{def}}$ to Σ_G enables to produce the formula $\neg Val(\alpha)$ since there is no attack from β or its source.

6.1.3 Reinstatement

We keep the original principle with the new notion of defence. If a semantics satisfies the reinstatement principle, then if an extension S defends an argument a, a must belong to S.

So we propose to reformulate the reinstatement property as follows:

$\mathbf{P}^{\mathbf{arg}}_{\mathbf{reins}}$**:** An argument must be accepted provided that, for each attack against it, the source or the attack itself is in turn attacked by a valid attack from an accepted argument (***i.e.*** an active attack).

The property $\mathbf{P}^{\mathbf{arg}}_{\mathbf{reins}}$ is expressed by the following formula:

$\mathbf{P}^{\mathbf{arg}}_{\mathbf{reins}}$ $(\forall c \in Argument)(((\forall \alpha \in Attack)(t_\alpha = c \rightarrow (\exists \beta \in Attack)(t_\beta \in \{s_\alpha, \alpha\} \wedge Val(\beta) \wedge Acc(s_\beta)))) \rightarrow Acc(c))$

Similarly we consider a kind of reinstatement principle for attacks.

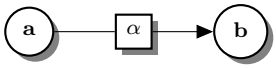
$\mathbf{P}^{\mathbf{att}}_{\mathbf{reins}}$ An attack may be valid provided that for each attack against it, either the source or the attack itself is in turn attacked by a valid attack from an accepted argument.

The property $\mathbf{P}^{\mathbf{att}}_{\mathbf{reins}}$ is expressed by the following formula:

$\mathbf{P}^{\mathbf{att}}_{\mathbf{reins}}$ $(\forall \delta \in Attack)(((\forall \alpha \in Attack)(t_\alpha = \delta \rightarrow (\exists \beta \in Attack)(t_\beta \in \{s_\alpha, \alpha\} \wedge Val(\beta) \wedge Acc(s_\beta)))) \rightarrow Val(\delta))$

The formulae that are obtained from $\mathbf{P}^{\mathbf{arg}}_{\mathbf{reins}}$ and $\mathbf{P}^{\mathbf{att}}_{\mathbf{reins}}$ are added to the base $\Sigma(G)$, thus producing the base $\Sigma_r(G)$.

A particular case: Consider the following graph:



Adding $\mathbf{P}^{\mathbf{arg}}_{\mathbf{reins}}$ to $\Sigma(G)$ enables to produce the formula $Acc(a)$ since a is unattacked.
Adding $\mathbf{P}^{\mathbf{att}}_{\mathbf{reins}}$ to $\Sigma(G)$ enables to produce the formula $Val(\alpha)$ since α is unattacked.

6.1.4 Stability

The stability property can be reformulated as follows:
If an argument is not accepted, it must be attacked by an active attack. The associated formula is:

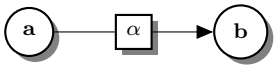
$\mathbf{P}^{\mathbf{arg}}_{\mathbf{sta}}$ $(\forall c \in Argument)(\neg Acc(c) \rightarrow (\exists \beta \in Attack)((t_\beta = c) \wedge Val(\beta) \wedge Acc(s_\beta)))$

Similarly we consider a kind of stability principle for attacks.
If an attack is not valid, it must be attacked by an active attack. The associated formula is:

$\mathbf{P}^{\mathbf{att}}_{\mathbf{sta}}$ $(\forall \alpha \in Attack)(\neg Val(\alpha) \rightarrow (\exists \beta \in Attack)((t_\beta = \alpha) \wedge Val(\beta) \wedge Acc(s_\beta)))$

The formulae that are obtained from $\mathbf{P}^{\mathbf{arg}}_{\mathbf{sta}}$ and $\mathbf{P}^{\mathbf{att}}_{\mathbf{sta}}$ are added to the base $\Sigma(G)$, thus producing the base $\Sigma_s(G)$.

A particular case: Consider the following graph:



Adding $\mathbf{P}^{\mathbf{arg}}_{\mathbf{sta}}$ to $\Sigma(G)$ enables to produce the formula $Acc(a)$ since a is unattacked.
Adding $\mathbf{P}^{\mathbf{att}}_{\mathbf{sta}}$ to $\Sigma(G)$ enables to produce the formula $Val(\alpha)$ since α is unattacked.

6.2 Definitions of semantics for an ASAF

Our purpose is to propose the definition of standard semantics for an ASAF. These definitions are guided by the principles presented above and should also extend standard semantics in the case of an AS.

6.2.1 The notion of structure

As expressed by Property $\mathbf{P}^{\mathbf{arg}}_{\mathbf{cf}}$ for instance, the fact that two arguments may be conflicting depends on the validity of the attack between them. So it would not be sound to give a definition of a set of arguments S being conflict-free, independently of a set of attacks. More generally, we need to reason about pairs of sets of arguments and sets of attacks, called "structures" in the following.

Def. 22 *Given ASAF =* $\langle \mathbf{A}, \mathbf{R}, s, t \rangle$*, a structure on ASAF is a pair* $U = (S, \Gamma)$ *such that* $S \subseteq \mathbf{A}$ *and* $\Gamma \subseteq \mathbf{R}$.

Intuitively, we are interested in structures $U = (S, \Gamma)$ such that S contains arguments that are accepted "owing to" U and Γ contains attacks which are valid "owing to" U. The precise meaning of "owing to" will depend on the considered semantics.

6.2.2 Conflict-free structures

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

Accordingly, we propose the following definitions:

Def. 23 *Given ASAF =* $\langle \mathbf{A}, \mathbf{R}, s, t \rangle$*, and a structure* $U = (S, \Gamma)$ *on ASAF.*
Let $a \in \mathbf{A}$ *and* $\alpha \in \mathbf{R}$.

- a *is defeated wrt* (S, Γ) *iff* $\exists \beta \in \Gamma$ *such that* $s(\beta) \in S$ *and* $t(\beta) = a$
- α *is inhibited wrt* (S, Γ) *iff* $\exists \beta \in \Gamma$ *such that* $s(\beta) \in S$ *and* $t(\beta) = \alpha$

The idea is that a structure is weakly conflict-free if no argument (resp. attack) of the structure is defeated (resp. inhibited) wrt the structure.

Def. 24 *Given ASAF* $\langle \mathbf{A}, \mathbf{R}, s, t \rangle$*, and a structure* $U = (S, \Gamma)$ *on ASAF.*
The structure (S, Γ) *is weakly conflict-free iff*

1. $\nexists a \in S$ *such that* a *is defeated wrt* (S, Γ)
2. $\nexists \alpha \in \Gamma$ *such that* α *is inhibited wrt* (S, Γ)

It is easy to prove that the structure $U = (S, \Gamma)$ is weakly conflict-free iff

1. $\forall a, b \in S, \nexists \alpha \in \Gamma$ such that $s(\alpha) = a$ and $t(\alpha) = b$
2. $\forall \beta \in \Gamma, \nexists \alpha \in \Gamma$ such that $s(\alpha) \in S$ and $t(\alpha) = \beta$

The following proposition is obvious:

Prop. 27 *Let* $S' \subseteq S \subseteq \mathbf{A}$ *and* $\Gamma' \subseteq \Gamma \subseteq \mathbf{R}$.
If the structure (S, Γ) *is weakly conflict-free, the structures* (S', Γ)*,* (S, Γ') *and* (S', Γ') *are also weakly conflict-free.*

Note that for any $\Gamma \subseteq \mathbf{R}$, the structure $(\varnothing, \Gamma)$ is weakly conflict-free. The same holds for classic AS, where an empty set of arguments is always conflict-free. Moreover for any $S \subseteq \mathbf{A}$ the structure $(S, \varnothing)$ is weakly conflict-free. The weak conflict-freeness requirement can be strengthened by adding attacks to Γ, namely the attacks which are not attacked. That leads to conflict-free structures.

Def. 25 *Given an ASAF* $\langle \mathbf{A}, \mathbf{R}, s, t \rangle$*, and a structure* $U = (S, \Gamma)$ *on ASAF.*
The structure (S, Γ) *is conflict-free iff*

1. *It is weakly conflict-free*
2. $\{\alpha \in \mathbf{R} | \nexists \beta \in \mathbf{R}$ *such that* $t(\beta) = \alpha\} \subseteq \Gamma$

Note that each conflict-free structure must contain all the unattacked attacks. Moreover, $\forall \alpha = (a, b)$, there is no conflict-free structure $U = (S, \Gamma)$ such that $a, b \in S$ and $\alpha \in \Gamma$. Similarly, $\forall \beta = (c, \alpha)$, there is no conflict-free structure $U = (S, \Gamma)$ such that $c \in S$ and $\alpha, \beta \in \Gamma$.

Note also that Proposition 27 does not hold for conflict-free structures. We only have the following result: If the structure (S, Γ) is conflict-free, for any $S' \subseteq S$ the structure (S', Γ) is also conflict-free.

In the case of an ASAF with simple (*i.e.* non recursive) attacks, as no attack is attacked, every conflict-free structure $U = (S, \Gamma)$ satisfies $\Gamma = \mathbf{R}$. As a consequence, the notion of conflict-free structure enables to recover Dung's conflict-free extensions, as shown below:

Def. 26 *Given an ASAF* $\langle \mathbf{A}, \mathbf{R}, s, t \rangle$*, and* $S \subseteq \mathbf{A}$.
S *is a conflict-free extension of ASAF iff there is* $\Gamma \subseteq \mathbf{R}$ *such that* (S, Γ) *is a conflict-free structure of ASAF.*

It is easy to prove that:

Prop. 28 *Given AS =* $\langle \mathbf{A}, \mathbf{R}, s, t \rangle$ *an ASAF with simple attacks, and* $S \subseteq \mathbf{A}$*,* S *is a conflict-free extension of AS in the sense of Def. 26 iff* S *is a conflict-free extension of* $\mathbf{A}$ *according to Dung's definition.*

Ex. 1 *The following structures are conflict-free:*

- $(\{a\}, \{\alpha\})$
- $(\{b\}, \{\alpha\})$
- $(\varnothing, \{\alpha\})$

Indeed, the conflict-free extensions of $\mathbf{A}$ *(in Dung's sense) are* $\{a\}$*,* $\{b\}$ *and* $\varnothing$*.*
However, the structures $(\{a, b\}, \varnothing)$ *and* $(\varnothing, \varnothing)$ *are only weakly conflict-free.*

Ex. 3 *The following structures are conflict-free:*

- $(\{a, c\}, \{\alpha, \beta\})$
- $(\{b\}, \{\alpha, \beta\})$
- $(\varnothing, \{\alpha, \beta\})$

However, the structures $(\{a, b, c\}, \varnothing)$*,* $(\{a, b\}, \{\beta\})$*,* $(\{b, c\}, \{\alpha\})$ *and* $(\varnothing, \varnothing)$ *are only weakly conflict-free.*

Ex. 14 *The following structures are conflict-free:*

- $(\{a, b, c\}, \{\beta\})$
- $(\{a\}, \{\alpha, \beta\})$

However, the structures $(\{a, c\}, \{\alpha\})$ *and* $(\{b, c\}, \{\alpha\})$ *are only weakly conflict-free.*
The structure $(\{a, c\}, \{\alpha, \beta\})$ *is not even weakly conflict-free.*

Ex. 21 *The following structures are conflict-free:*

- $(\{a, b, c\}, \{\delta\})$
- $(\{a, c\}, \{\alpha, \delta\})$
- $(\{a, c\}, \{\beta, \delta\})$
- $(\{a\}, \{\alpha, \beta, \delta\})$

Ex. 25 *The weakly conflict-free structures are :*

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

They are all conflict-free too.

6.2.3 Admissible structures

As done for conflict-freeness, the definition of an argument a being acceptable wrt a set of arguments S should be relative to a set of attacks. That leads to consider a notion of admissibility for structures.

Def. 27 *Given ASAF =* $\langle \mathbf{A}, \mathbf{R}, s, t \rangle$*, and a structure* $U = (S, \Gamma)$ *on ASAF.*
Let $a \in \mathbf{A}$ *and* $\alpha \in \mathbf{R}$*.*

- a *is acceptable wrt* U *iff* $\forall \beta \in \mathbf{R}$ *such that* $t(\beta) = a$*, either* β *is inhibited wrt* U *or* $s(\beta)$ *is defeated wrt* U*.*
- α *is acceptable wrt* U *iff* $\forall \beta \in \mathbf{R}$ *such that* $t(\beta) = \alpha$*, either* β *is inhibited wrt* U *or* $s(\beta)$ *is defeated wrt* U*.*

In other words, $x \in \mathbf{A} \cup \mathbf{R}$ is acceptable wrt a structure $U = (S, \Gamma)$ iff for each attack $\beta \in \mathbf{R}$ such that $t(\beta) = x$, there exists $\gamma \in \Gamma$ with $s(\gamma) \in S$ and $t(\gamma) = \{\beta, s(\beta)\}$.

Then admissible structures can be defined as follows:

Def. 28 *Given = ASAF* $\langle \mathbf{A}, \mathbf{R}, s, t \rangle$*, and a structure* $U = (S, \Gamma)$ *on ASAF.*

1. *The structure U is weakly admissible iff U is weakly conflict-free and $\forall x \in (S \cup \Gamma)$, x is acceptable wrt U.*

2. *The structure U is admissible iff U is conflict-free and $\forall x \in (S \cup \Gamma)$, x is acceptable wrt U.*

Note that $\forall \alpha = (a, b)$, if α and a are unattacked, there is no admissible structure $U = (S, \Gamma)$ such that $b \in S$. Similarly, $\forall \beta = (c, \alpha)$, if β and c are unattacked, there is no admissible structure $U = (S, \Gamma)$ such that $\alpha \in \Gamma$.

Admissible structures enable to recover Dung's admissible extensions, as shown below:

Def. 29 *Given an ASAF $\langle \mathbf{A}, \mathbf{R}, s, t \rangle$, and $S \subseteq \mathbf{A}$.*
S is an admissible extension of ASAF iff there is $\Gamma \subseteq \mathbf{R}$ such that (S, Γ) is an admissible structure of ASAF.

Prop. 29 *Given AS = $\langle \mathbf{A}, \mathbf{R}, s, t \rangle$ an ASAF with simple attacks. Let $a \in \mathbf{A}$.*

- *If a is acceptable wrt a structure $U = (S, \Gamma)$ on AS, then a is acceptable wrt S in Dung's sense.*
- *If a is acceptable wrt S in Dung's sense, then a is acceptable wrt the structure $(S, \mathbf{R})$.*

Prop. 30 *Given AS = $\langle \mathbf{A}, \mathbf{R}, s, t \rangle$ an ASAF with simple attacks, and $S \subseteq \mathbf{A}$, S is an admissible extension of AS (in the sense of Def. 29) iff S is an admissible subset of $\mathbf{A}$ according to Dung's definition of admissibility.*

Ex. 1

- *$(\{a, b\}, \varnothing)$ is not weakly-admissible, since b is not acceptable wrt this structure*
- *$(\{b\}, \{\alpha\})$ is conflict-free, but not admissible for the same reason (b is not acceptable wrt this structure)*
- *The admissible structures are $(\varnothing, \{\alpha\})$ and $(\{a\}, \{\alpha\})$*

Ex. 3 *The admissible structures are $(\varnothing, \{\alpha, \beta\})$, $(\{a\}, \{\alpha, \beta\})$, and $(\{a, c\}, \{\alpha, \beta\})$.*

Ex. 14 *No admissible structure contains α. The admissible structures are:*
$(\varnothing, \{\beta\})$, $(\{a\}, \{\beta\})$, $(\{c\}, \{\beta\})$, $(\{b, c\}, \{\beta\})$, $(\{a, c\}, \{\beta\})$ and $(\{a, b, c\}, \{\beta\})$.

Ex. 21 *Any admissible structure (S, Γ) is such that $\delta \in \Gamma$.*
Assume there exists an admissible structure $U = (S, \Gamma)$ such that $b \in S$. Then α must be inhibited by U, so $c \in S$ and $\beta \in \Gamma$. Then β must be acceptable wrt U, so b must be defeated by U. That implies that $a \in S$ and $\alpha \in \Gamma$. However, the structure $(\{a, b, c\}, \{\alpha, \beta, \delta\})$ is not even weakly conflict-free. So no admissible structure may contain b.
Similarly, it is easy to prove that no admissible structure may contain α (resp. β).
The admissible structures are $(\varnothing, \{\delta\})$, $(\{a\}, \{\delta\})$, $(\{c\}, \{\delta\})$ and $(\{a, c\}, \{\delta\})$.

Ex. 25 *Assume there exists an admissible structure $U = (S, \Gamma)$ such that $\alpha \in \Gamma$. Then α must be acceptable wrt U, so S must contain a. However, the structure $(\{a\}, \{\alpha\})$ is not conflict-free.*
So the admissible structures are$(\varnothing, \varnothing)$ and $(\{a\}, \varnothing)$.

6.2.4 Complete structures

Following the definitions of standard semantics in the classic case, we define complete structures as admissible structures which contain all the arguments (resp. attacks) that are acceptable wrt the structure.

Def. 30 *Given ASAF = $\langle \mathbf{A}, \mathbf{R}, s, t \rangle$, and a structure $U = (S, \Gamma)$ on ASAF.*
The structure U is complete iff U is admissible and $\forall x \in \mathbf{A}$ (resp. $x \in \mathbf{R}$), if x is acceptable wrt U then $x \in S$ (resp. $x \in \Gamma$).

Def. 31 *Given an ASAF $\langle \mathbf{A}, \mathbf{R}, s, t \rangle$, and $S \subseteq \mathbf{A}$.*
S is a complete extension of ASAF iff there is $\Gamma \subseteq \mathbf{R}$ such that (S, Γ) is a complete structure of ASAF.

Note that each complete structure must contain all the unattacked arguments and all the unattacked attacks.

Prop. 31 *Given AS = $\langle \mathbf{A}, \mathbf{R}, s, t \rangle$ an ASAF with simple attacks, and $S \subseteq \mathbf{A}$, S is a complete extension of AS (in the sense of Def. 31) iff S is a complete extension of $\mathbf{A}$ according to Dung's definition.*

Ex. 1 *There is only one complete structure* $(\{a\}, \{\alpha\})$.

Ex. 3 *There is only one complete structure* $(\{a, c\}, \{\alpha, \beta\})$.

Ex. 14 *There is only one complete structure* $(\{a, b, c\}, \{\beta\})$.

Ex. 21 *There is only one complete structure* $(\{a, c\}, \{\delta\})$.

Ex. 25 *There is only one complete structure* $(\{a\}, \varnothing)$.

6.2.5 Stable structures

In the classic case, stable extensions are defined as conflict-free extensions that attack external arguments. It can be proved that stable extensions are also admissible and even complete extensions. So we should provide a definition of stable structures that preserves the same relation between semantics.

Def. 32 *Given ASAF = $\langle \mathbf{A}, \mathbf{R}, s, t \rangle$, and a structure $U = (S, \Gamma)$ on ASAF.*
The structure U is stable iff

1. *U is conflict-free*
2. *$\forall a \notin S$, a is defeated wrt U*
3. *$\forall \alpha \notin \Gamma$, α is inhibited wrt U*

Def. 33 *Given an ASAF $\langle \mathbf{A}, \mathbf{R}, s, t \rangle$, and $S \subseteq \mathbf{A}$.*
S is a stable extension of ASAF iff there is $\Gamma \subseteq \mathbf{R}$ such that (S, Γ) is a stable structure of ASAF.

Prop. 32 *Given AS = $\langle \mathbf{A}, \mathbf{R}, s, t\rangle$ an ASAF with simple attacks, and $S \subseteq \mathbf{A}$, S is a stable extension of AS (in the sense of Def. 33 on the previous page) iff S is a stable extension of $\mathbf{A}$ according to Dung's definition.*

Ex. 23

$(\{a, b, c\}, \varnothing)$ is the unique complete structure. However it is not stable, as no attack is inhibited by this structure.

Prop. 33 *Let ASAF $= \langle \mathbf{A}, \mathbf{R}, s, t\rangle$. Stable structures of ASAF are also complete structures of ASAF.*

6.3 Characterizing semantics for an ASAF

Our purpose is to propose characterizations of the structures in different semantics in terms of models of the bases $\Sigma(G)$, $\Sigma_d(G)$, $\Sigma_r(G)$, $\Sigma_s(G)$.

Remark:
The Herbrand models of the base $\Sigma(G)$ representing an ASAF contain literals of the form $Acc(x)$, $NAcc(y)$, and also literals of the form $Val(\alpha)$, $Gr(\beta)$ and $Act(\gamma)$. Both types of literals should be exploited. However, as explained before, the literals of the form $Gr(\alpha)$ are not informative since $Gr(\alpha)$ is *true* if and only if $Acc(s_\alpha)$ is *true* (due to $\mathbf{P_{gr}}$). So we could simplify the representation of a Herbrand model of $\Sigma(G)$ by omitting the literals of the form $Gr(\alpha)$. Similarly, due to Property $\mathbf{P_{act}}$, we could omit the literals of the form $Act(\alpha)$.
In the following, a Herbrand model $\mathcal{I}$ of $\Sigma(G)$ will be represented by the set of the positive literals of the form $Acc(x)$, $NAcc(x)$, $Val(\alpha)$ that are *true* in $\mathcal{I}$.

Formally, we define:

Def. 34 *Let $\mathcal{I}$ be an interpretation of $\Sigma(G)$.*

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

The following results hold.

First the characterization of weakly conflict-free structures is given by the following result:

Prop. 34 *Given an ASAF $\langle \mathbf{A}, \mathbf{R}, s, t\rangle$ represented by the base $\Sigma(G)$, $S \subseteq \mathbf{A}$ and $\Gamma \subseteq \mathbf{R}$.*
The structure (S, Γ) is weakly conflict-free iff there exists $\mathcal{I}$ model of $\Sigma(G)$ such that $S_\mathcal{I} = S$ and $\Gamma_\mathcal{I} = \Gamma$.

Then the characterization of weakly admissible structures is given by the following result:

Prop. 35 *Given an ASAF $\langle \mathbf{A}, \mathbf{R}, s, t\rangle$ represented by the base $\Sigma(G)$, $S \subseteq \mathbf{A}$ and $\Gamma \subseteq \mathbf{R}$.*
The structure (S, Γ) is weakly admissible iff there exists $\mathcal{I}$ model of $\Sigma_d(G)$ such that $S_\mathcal{I} = S$ and $\Gamma_\mathcal{I} = \Gamma$.

The characterization of complete structures is given by the following result:

Prop. 36 *Given an ASAF* $\langle \mathbf{A}, \mathbf{R}, s, t\rangle$ *represented by the base* $\Sigma(G)$, $S \subseteq \mathbf{A}$ *and* $\Gamma \subseteq \mathbf{R}$. *The structure* (S, Γ) *is complete iff there exists* $\mathcal{I}$ *model of* $\Sigma_d(G) \cup \Sigma_r(G)$ *such that* $S_{\mathcal{I}} = S$ *and* $\Gamma_{\mathcal{I}} = \Gamma$.

And finally the characterization of stable structures is given by the following result:

Prop. 37 *Given an ASAF* $\langle \mathbf{A}, \mathbf{R}, s, t\rangle$ *represented by the base* $\Sigma(G)$, $S \subseteq \mathbf{A}$ *and* $\Gamma \subseteq \mathbf{R}$. *The structure* (S, Γ) *is stable iff there exists* $\mathcal{I}$ *model of* $\Sigma_s(G)$ *such that* $S_{\mathcal{I}} = S$ *and* $\Gamma_{\mathcal{I}} = \Gamma$.

Some concluding remarks

- In order to get a characterization of conflict-free structures (resp. admissible structures), we must add a formula to the base $\Sigma(G)$ (resp. $\Sigma_d(G)$). This formula expresses the fact that each attack which is not attacked must be valid.
  Note that this formula is entailed by $\Sigma_r(G)$.

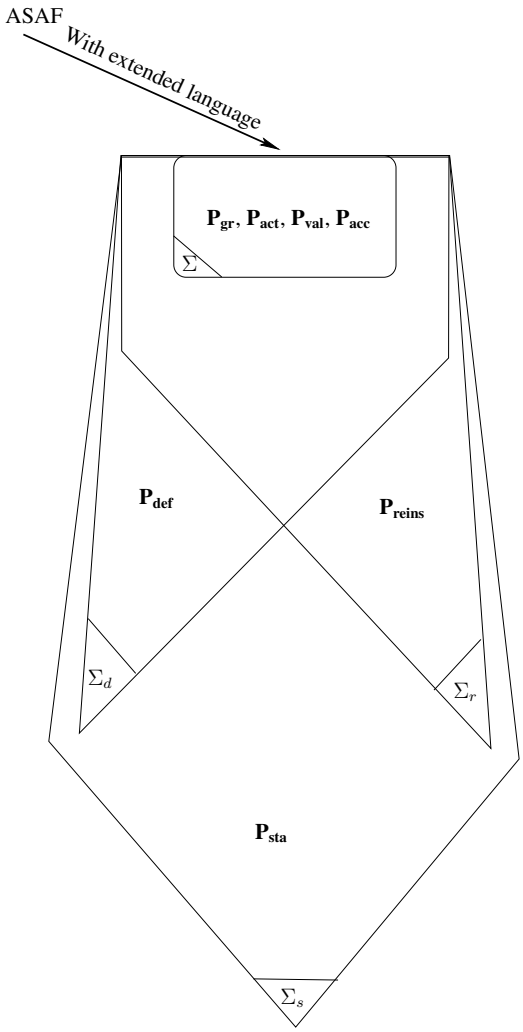
- Once interesting structures have been obtained, for instance the complete structures, one could be interested in the active attacks determined by these structures.

  Due to the definition of an active attack, we should define these attacks as follows:
  Let $U = (S, \Gamma)$ be a structure on ASAF, $Act(U) = \{\alpha \in \Gamma | s(\alpha) \in S\}$.

  This notion of active attacks determined by a structure will enable us to draw comparisons with the work of [6] and [22].

6.4 Synthesis for ASAF

The following figure describes the different encodings that can be used for an ASAF.


ASAF
With extended language
$\mathbf{P_{gr}}$, $\mathbf{P_{act}}$, $\mathbf{P_{val}}$, $\mathbf{P_{acc}}$
Σ
$\mathbf{P_{def}}$
$\mathbf{P_{reins}}$
Σ_d
Σ_r
$\mathbf{P_{sta}}$
Σ_s

Chapter 7

Future works

Beyond the open questions raised in Section 6.2 on page 54, it is mandatory to pursue this work towards the following directions:

- In the case of an AS:
  1. Discuss the related works in logic programming, ASP, labelling-based argumentation frameworks, Abstract Dialectical Frameworks (ADF).
  2. Implement the different procedures (searching for models, for extensions).
- In the case of a ASAF, draw a comparative study with the work of [6, 21, 15] which propose to encode recursive attacks with different frameworks (AFRA, ASAF and Meta Argumentation Systems – MAS).

In a longer-term perspective, it would be of interest to extend this work to the case of bipolar argumentation frameworks, including supports and attacks between arguments, both interactions being possibly recursive.

Bibliography

[1] J. M. Alliot, R. Demolombe, L. Fariñas del Cerro, M. Diéguez, and N. Obeid. Reasoning on molecular interaction maps. In *Proc. of ESCIM*, pages 263–269, 2015. Available at:http://escim2015.uca.es/wp-content/uploads/2015/02/ESCIM2015-FINAL.pdf.

[2] L. Amgoud, P. Besnard, and A. Hunter. Logical representation and analysis for rc-arguments. In *Proc. of ICTAI*, pages 104–110, 2015.

[3] L. Amgoud and C. Cayrol. A reasoning model based on the production of acceptable arguments. *Annals of Mathematics and Artificial Intelligence*, 34:197–216, 2002.

[4] L. Amgoud, N. Maudet, and S. Parsons. Modelling dialogues using argumentation. In *Proc. of ICMAS*, pages 31–38, 2000.

[5] O. Arieli and M.W.A. Caminada. A QBF-based formalization of abstract argumentation semantics. *Journal of Applied Logic*, 11(2):229 – 252, 2013.

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

[7] 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*, pages 241–248, 2015.

[8] P. Besnard and S. Doutre. Checking the acceptability of a set of arguments. In *Proc. of NMR*, pages 59–64, 2004.

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

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

[11] B. Bogaerts, T. Janhunen, and S. Tasharrofi. Declarative solver development: Case studies. In *Proc. of KR*, pages 74–83, 2016.

[12] G. Brewka and S. Woltran. Abstract dialectical frameworks. In *Proc. of KR*, pages 102–111, 2010.

[13] M. Caminada, S. Sá, J. Alcântara, and W. Dvořák. On the equivalence between logic programming semantics and argumentation semantics. *IJAR*, 58:87 – 111, 2015.

[14] J. L. Carballido, J. C. Nieves, and M. Osorio. Inferring preferred extensions by pstable semantics. *Inteligencia artificial: Revista Iberoamericana de Inteligencia Artificial*, 13(41):38–53, 2009.

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

[16] C. Cayrol, L. Fariñas del Cerro, and M-C. Lagasquie-Schiex. A logical vision of abstract argumentation systems with bipolar and recursive interactions. Technical Report RR- -2016- -02- -FR, IRIT, 2016.

[17] C. Cayrol and M-C. Lagasquie-Schiex. On the acceptability of arguments in bipolar argumentation frameworks. In *Proc. of ECSQARU*, pages 378–389, 2005.

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

[19] C. Cayrol and M-C. Lagasquie-Schiex. An axiomatic approach to support in argumentation. In *Proc. of TAFA (LNAI 9524, revised selected papers)*, pages 74–91, 2015.

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

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

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

[23] R. Demolombe, L. Fariñas del Cerro, and N. Obeid. Molecular Interaction Automated Maps (regular paper). In *Proc. of LNMR*, pages 31–42, 2013.

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

[25] 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*, 75:57 – 92, 2016.

[26] D. Grossi. On the logic of argumentation theory. In *Proc. of AAMAS*, pages 409–416, 2010.

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

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

[29] S. Modgil. Reasoning about preferences in argumentation frameworks. *Artif. Intell.*, 173:901–934, 2009.

[30] F. Nouioua. AFs with necessities: further semantics and labelling characterization. In *Proc. of SUM*, pages 120–133, 2013.

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

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

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

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

[35] M. Osorio, J. C. Nieves, and A. Santoyo. Complete extensions as clark's completion semantics. In *Proc. of the Mexican International Conference on Computer Science*, pages 81–88, 2013.

[36] S. Polberg and N. Oren. Revisiting support in abstract argumentation systems. In *Proc. of COMMA*, pages 369–376, 2014.

[37] H. Prakken. On support relations in abstract argumentation as abstraction of inferential relations. In *Proc. of ECAI*, pages 735–740, 2014.

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

[39] J. P. Wallner, G. Weissenbacher, and S. Woltran. Advanced SAT techniques for abstract argumentation. In *Proc. of CLIMA*, pages 138–154, 2013.

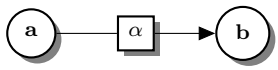
Appendix A

Description of the examples

A.1 Examples without recursivity (AS)

A.1.1 Example 1

Ex. 1



$\Sigma^0(G)$	$= \{$	$Acc(a) \rightarrow NAcc(b)$ $NAcc(b) \rightarrow \neg Acc(b)$	$\}$
$\Sigma(G)$	$= \{$	$Gr(\alpha) \leftrightarrow Acc(a)$ $Act(\alpha) \leftrightarrow (Gr(\alpha) \wedge Val(\alpha))$ $(Val(\alpha) \wedge Acc(a)) \rightarrow NAcc(b)$ $NAcc(b) \rightarrow \neg Acc(b)$	$\}$
$\Sigma_{as}(G)$	$= \Sigma(G) \cup \{$	$Val(\alpha)$	$\}$
$\Sigma_d(G)$	$= \Sigma_{as}(G) \cup \{$	$\neg Acc(b)$	$\}$
$\Sigma_r(G)$	$= \Sigma_{as}(G) \cup \{$	$Acc(a)$	$\}$
$\Sigma_s(G)$	$= \Sigma_{as}(G) \cup \{$	$Acc(a)$ $\neg Acc(b) \rightarrow Acc(a)$	$\}$

Among the logical consequences of $\Sigma^0(G)$ *is the formula:*

$Acc(a) \rightarrow \neg Acc(b)$

It follows that if a *is accepted then* b *is not accepted.*

Among the logical consequences of $\Sigma_{as}(G)$ *are the formulae:*

$Val(\alpha)$

$Act(\alpha) \leftrightarrow Gr(\alpha)$ *and so* $Act(\alpha) \leftrightarrow Acc(a)$

$Acc(a) \rightarrow \neg Acc(b)$ *and so* $Act(\alpha) \rightarrow \neg Acc(b)$

It follows that α *is active iff* a *is accepted and if* α *is active then* b *is not accepted..*

Note that $\neg Acc(b)$ *follows from* $\Sigma_r(G)$.

The following table displays the standard Dung extensions of $\mathcal{G}$:

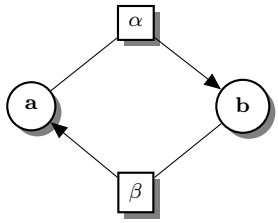
Semantics	*Extensions*
Grounded	$\{a\}$
Preferred	$\{a\}$
Stable	$\{a\}$
Complete	$\{a\}$

The following table shows the sets $S_{\mathcal{I}} = \{x \in \mathbf{A} | \mathcal{I}(Acc(x)) = \textit{true}\}$ associated to the models $\mathcal{I}$ of some bases:

Base	$S_{\mathcal{I}}$
$\Sigma(G)$	$\varnothing$ $\{a\}$ $\{b\}$
$\Sigma_d(G)$	$\varnothing$ $\{a\}$
$\Sigma_r(G)$	$\{a\}$

A.1.2 Example 2

Ex. 2



$\Sigma^0(G)$	$= \{$	$Acc(a) \rightarrow NAcc(b)$ $Acc(b) \rightarrow NAcc(a)$ $NAcc(a) \rightarrow \neg Acc(a)$ $NAcc(b) \rightarrow \neg Acc(b)$	$\}$
$\Sigma(G)$	$= \{$	$Gr(\alpha) \leftrightarrow Acc(a)$ $Gr(\beta) \leftrightarrow Acc(b)$ $Act(\alpha) \leftrightarrow (Gr(\alpha) \wedge Val(\alpha))$ $Act(\beta) \leftrightarrow (Gr(\beta) \wedge Val(\beta))$ $(Val(\alpha) \wedge Acc(a)) \rightarrow NAcc(b)$ $NAcc(b) \rightarrow \neg Acc(b)$ $(Val(\beta) \wedge Acc(b)) \rightarrow NAcc(a)$ $NAcc(a) \rightarrow \neg Acc(a)$	$\}$
$\Sigma_{as}(G)$	$= \Sigma(G) \cup \{$	$Val(\alpha), Val(\beta)\}$	$\}$
$\Sigma_d(G)$	$= \Sigma_{as}(G)$		
$\Sigma_r(G)$	$= \Sigma_{as}(G)$		
$\Sigma_s(G)$	$= \Sigma_{as}(G) \cup \{$	$\neg Acc(a) \rightarrow Acc(b)$ $\neg Acc(b) \rightarrow Acc(a)$	$\}$

Among the logical consequences of $\Sigma^0(G)$ *is the formula:*

$\neg(Acc(a) \wedge Acc(b))$.

It follows that a *and* b *cannot be jointly accepted.*

Among the logical consequences of $\Sigma_{as}(G)$ *are the formulae:*

$Val(\alpha)$ *and* $Val(\beta)$

$Act(\alpha) \leftrightarrow Gr(\alpha)$ *and* $Act(\beta) \leftrightarrow Gr(\beta)$ *and so* $Act(\alpha) \leftrightarrow Acc(a)$ *and* $Act(\beta) \leftrightarrow Acc(b)$

$\neg(Acc(a) \wedge Acc(b))$ *and so* $Act(\alpha) \rightarrow \neg Acc(b)$ *and* $Act(\beta) \rightarrow \neg Acc(a)$

It follows that α *is active iff* a *is accepted,* β *is active iff* b *is accepted,* α *et* β *cannot be jointly active, and* a *and* b *cannot be jointly accepted.*

The following table displays the standard Dung extensions of $\mathcal{G}$*:*

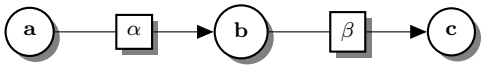
Semantics	*Extensions*
Grounded	$\varnothing$
Preferred	$\{a\}$ $\{b\}$
Stable	$\{a\}$ $\{b\}$
Complete	$\varnothing$ $\{a\}$ $\{b\}$

The following table shows the sets $S_{\mathcal{I}} = \{x \in \mathbf{A} | \mathcal{I}(Acc(x)) = \textit{true}\}$ associated to the models $\mathcal{I}$ of some bases:

Base	$S_{\mathcal{I}}$
$\Sigma(G)$	$\varnothing$ $\{a\}$ $\{b\}$
$\Sigma_d(G)$	$\varnothing$ $\{a\}$ $\{b\}$
$\Sigma_r(G)$	$\varnothing$ $\{a\}$ $\{b\}$

A.1.3 Example 3

Ex. 3



$\Sigma^0(G)$	$= \{$	$Acc(a) \rightarrow NAcc(b)$ $Acc(b) \rightarrow NAcc(c)$ $NAcc(b) \rightarrow \neg Acc(b)$ $NAcc(c) \rightarrow \neg Acc(c)$	$\}$
$\Sigma(G)$	$= \{$	$Gr(\alpha) \leftrightarrow Acc(a)$ $Gr(\beta) \leftrightarrow Acc(b)$ $Act(\alpha) \leftrightarrow (Gr(\alpha) \wedge Val(\alpha))$ $Act(\beta) \leftrightarrow (Gr(\beta) \wedge Val(\beta))$ $(Val(\alpha) \wedge Acc(a)) \rightarrow NAcc(b)$ $NAcc(b) \rightarrow \neg Acc(b)$ $(Val(\beta) \wedge Acc(b)) \rightarrow NAcc(c)$ $NAcc(c) \rightarrow \neg Acc(c)$	$\}$
$\Sigma_{as}(G)$	$= \Sigma(G) \cup \{$	$Val(\alpha)$ $Val(\beta)$	$\}$
$\Sigma_{d}(G)$	$= \Sigma_{as}(G) \cup \{$	$\neg Acc(b)$ $Acc(c) \rightarrow Acc(a)$	$\}$
$\Sigma_{r}(G)$	$= \Sigma_{as}(G) \cup \{$	$Acc(a)$ $Acc(a) \rightarrow Acc(c)$	$\}$
$\Sigma_{s}(G)$	$= \Sigma_{as}(G) \cup \{$	$Acc(a)$ $\neg Acc(b) \rightarrow Acc(a)$ $\neg Acc(c) \rightarrow Acc(b)$	$\}$

Among the logical consequences of $\Sigma^0(G)$ *are the formulae:*

$\neg(Acc(a) \wedge Acc(b))$ *et* $\neg(Acc(c) \wedge Acc(b))$

It follows that if a *is accepted then* b *is not accepted. Note that* c *could be accepted.*

Among the logical consequences of $\Sigma_{as}(G)$ *are the formulae:*

$Val(\alpha)$ *et* $Val(\beta)$

$Act(\alpha) \leftrightarrow Gr(\alpha)$ *and* $Act(\beta) \leftrightarrow Gr(\beta)$ *and so* $Act(\alpha) \leftrightarrow Acc(a)$ *and* $Act(\beta) \leftrightarrow Acc(b)$

$\neg(Acc(a) \wedge Acc(b))$ *and* $\neg(Acc(c) \wedge Acc(b))$ *and so*

$Act(\alpha) \rightarrow \neg Acc(b)$ *and* $Act(\beta) \rightarrow \neg Acc(c)$

It follws that α *is active iff* a *is accepted,* β *is active iff* b *is accepted, if* α *is active* b *is not accepted and if* β *is active then* c *is not accepted.*

Note that $Acc(c)$ *and* $\neg Acc(b)$ *follow from* $\Sigma_r(G)$.

The following table displays the standard Dung extensions of $\mathcal{G}$*:*

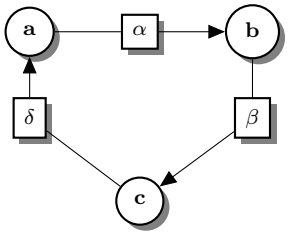
Semantics	*Extensions*
Grounded	$\{a, c\}$
Preferred	$\{a, c\}$
Stable	$\{a, c\}$
Complete	$\{a, c\}$

The following table shows the sets $S_{\mathcal{I}} = \{x \in \mathbf{A} | \mathcal{I}(Acc(x)) = \textit{true}\}$ associated to the models $\mathcal{I}$ of some bases:

Base	$S_{\mathcal{I}}$
$\Sigma(G)$	$\varnothing$ $\{a\}$ $\{b\}$ $\{c\}$ $\{a, c\}$
$\Sigma_d(G)$	$\varnothing$ $\{a\}$ $\{a, c\}$
$\Sigma_r(G)$	$\{a, c\}$

A.1.4 Example 4

Ex. 4



$\Sigma^0(G)$	$= \{$	$Acc(a) \rightarrow NAcc(b)$ $Acc(b) \rightarrow NAcc(c)$ $Acc(c) \rightarrow NAcc(a)$ $NAcc(a) \rightarrow \neg Acc(a)$ $NAcc(b) \rightarrow \neg Acc(b)$ $NAcc(c) \rightarrow \neg Acc(c)$	$\}$
$\Sigma(G)$	$= \{$	$Gr(\alpha) \leftrightarrow Acc(a)$ $Gr(\beta) \leftrightarrow Acc(b)$ $Gr(\delta) \leftrightarrow Acc(c)$ $Act(\alpha) \leftrightarrow (Gr(\alpha) \wedge Val(\alpha))$ $Act(\beta) \leftrightarrow (Gr(\beta) \wedge Val(\beta))$ $Act(\delta) \leftrightarrow (Gr(\delta) \wedge Val(\delta))$ $(Val(\alpha) \wedge Acc(a)) \rightarrow NAcc(b)$ $NAcc(b) \rightarrow \neg Acc(b)$ $(Val(\beta) \wedge Acc(b)) \rightarrow NAcc(c)$ $NAcc(c) \rightarrow \neg Acc(c)$ $(Val(\delta) \wedge Acc(c)) \rightarrow NAcc(a)$ $NAcc(a) \rightarrow \neg Acc(a)$	$\}$
$\Sigma_{as}(G)$	$= \Sigma(G) \cup \{$	$Val(\alpha)$ $Val(\beta)$ $Val(\delta)$	$\}$
$\Sigma_{d}(G)$	$= \Sigma_{as}(G) \cup \{$	$Acc(a) \rightarrow Acc(b)$ $Acc(b) \rightarrow Acc(c)$ $Acc(c) \rightarrow Acc(a)$	$\}$
$\Sigma_{r}(G)$	$= \Sigma_{as}(G) \cup \{$	$Acc(a) \rightarrow Acc(c)$ $Acc(b) \rightarrow Acc(a)$ $Acc(c) \rightarrow Acc(b)$	$\}$
$\Sigma_{s}(G)$	$= \Sigma_{as}(G) \cup \{$	$\neg Acc(a) \rightarrow Acc(c)$ $\neg Acc(b) \rightarrow Acc(a)$ $\neg Acc(c) \rightarrow Acc(b)$	$\}$

Among the logical consequences of $\Sigma^0(G)$ *are the formulae:*

$\neg(Acc(a) \wedge Acc(b))$, $\neg(Acc(b) \wedge Acc(c))$ *and* $\neg(Acc(a) \wedge Acc(c))$

It follows that a *and* b *(resp.* a *and* c*,* b*,* c*) cannot be jointly accepted.*

Among the logical consequences of $\Sigma_{as}(G)$ *are the formulae:*

$Val(\alpha)$, $Val(\beta)$ *and* $Val(\delta)$,
$Act(\alpha) \leftrightarrow Gr(\alpha)$, $Act(\beta) \leftrightarrow Gr(\beta)$ *and* $Act(\delta) \leftrightarrow Gr(\delta)$
$Act(\alpha) \leftrightarrow Acc(a)$, $Act(\beta) \leftrightarrow Acc(b)$ *and* $Act(\delta) \leftrightarrow Acc(c)$
$\neg(Acc(a) \wedge Acc(b))$, $\neg(Acc(b) \wedge Acc(c))$ *and* $\neg(Acc(a) \wedge Acc(c))$
and so $Act(\alpha) \rightarrow (\neg Acc(c) \wedge \neg Acc(b))$, $Act(\beta) \rightarrow (\neg Acc(a) \wedge \neg Acc(c))$
and $Act(\delta) \rightarrow (\neg Acc(b) \wedge \neg Acc(a))$

It follows that a *and* b *(resp.* a *and* c*,* b*,* c*) cannot be jointly accepted.*

The following table displays the standard Dung extensions of $\mathcal{G}$*:*

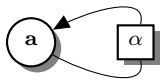
Semantics	*Extensions*
Grounded	$\varnothing$
Preferred	$\varnothing$
Stable	*none*
Complete	$\varnothing$

The following table shows the sets $S_{\mathcal{I}} = \{x \in \mathbf{A} | \mathcal{I}(Acc(x)) = \textit{true}\}$ *associated to the models* $\mathcal{I}$ *of some bases:*

Base	$S_{\mathcal{I}}$
$\Sigma(G)$	$\varnothing$ $\{a\}$ $\{b\}$ $\{c\}$
$\Sigma_d(G)$	$\varnothing$
$\Sigma_r(G)$	$\varnothing$

A.1.5 Example 5

Ex. 5



$\Sigma^0(G)$	$= \{$	$Acc(a) \rightarrow NAcc(a)$ $NAcc(a) \rightarrow \neg Acc(a)$	$\}$
$\Sigma(G)$	$= \{$	$Gr(\alpha) \leftrightarrow Acc(a)$ $Act(\alpha) \leftrightarrow (Gr(\alpha) \wedge Val(\alpha))$ $(Val(\alpha) \wedge Acc(a)) \rightarrow NAcc(a)$ $NAcc(a) \rightarrow \neg Acc(a)$	$\}$
$\Sigma_{as}(G)$	$= \Sigma(G) \cup \{$	$Val(\alpha)$	$\}$
$\Sigma_d(G)$	$= \Sigma_{as}(G)$		
$\Sigma_r(G)$	$= \Sigma_{as}(G)$		
$\Sigma_s(G)$	$= \Sigma_{as}(G) \cup \{$	$Acc(a)$	$\}$

Among the logical consequences of $\Sigma^0(G)$ *is the literal:*

$\neg Acc(a)$

It follows that a *cannot be accepted.*

Among the logical consequences of $\Sigma_{as}(G)$ *are the formulae:*

$Val(\alpha)$

$Act(\alpha) \leftrightarrow Gr(\alpha)$ *and* $Act(\alpha) \leftrightarrow Acc(a)$

$\neg Acc(a)$

It follows that a *cannot be accepted and that* α *is valid yet not active.*

The following table displays the standard Dung extensions of $\mathcal{G}$*:*

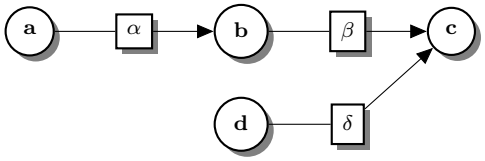
Semantics	*Extensions*
Grounded	$\varnothing$
Preferred	$\varnothing$
Stable	*none*
Complete	$\varnothing$

The following table shows the sets $S_{\mathcal{I}} = \{x \in \mathbf{A} | \mathcal{I}(Acc(x)) = \textit{true}\}$ *associated to the models* $\mathcal{I}$ *of some bases:*

Base	$S_{\mathcal{I}}$
$\Sigma(G)$	$\varnothing$
$\Sigma_d(G)$	$\varnothing$
$\Sigma_r(G)$	$\varnothing$

A.1.6 Example 6

Ex. 6



$\Sigma^0(G)$	$= \{$	$Acc(a) \rightarrow NAcc(b)$ $Acc(b) \rightarrow NAcc(c)$ $Acc(d) \rightarrow NAcc(c)$ $NAcc(b) \rightarrow \neg Acc(b)$ $NAcc(c) \rightarrow \neg Acc(c)$	$\}$
$\Sigma(G)$	$= \{$	$Gr(\alpha) \leftrightarrow Acc(a)$ $Gr(\beta) \leftrightarrow Acc(b)$ $Gr(\delta) \leftrightarrow Acc(d)$ $Act(\alpha) \leftrightarrow (Gr(\alpha) \wedge Val(\alpha))$ $Act(\beta) \leftrightarrow (Gr(\beta) \wedge Val(\beta))$ $Act(\delta) \leftrightarrow (Gr(\delta) \wedge Val(\delta))$ $(Val(\alpha) \wedge Acc(a)) \rightarrow NAcc(b)$ $NAcc(b) \rightarrow \neg Acc(b)$ $(Val(\beta) \wedge Acc(b)) \rightarrow NAcc(c)$ $NAcc(c) \rightarrow \neg Acc(c)$ $(Val(\delta) \wedge Acc(d)) \rightarrow NAcc(c)$	$\}$
$\Sigma_{as}(G)$	$= \Sigma(G) \cup \{$	$Val(\alpha)$ $Val(\beta)$ $Val(\delta)$	$\}$
$\Sigma_d(G)$	$= \Sigma_{as}(G) \cup \{$	$\neg Acc(b)$ $\neg Acc(c)$	$\}$
$\Sigma_r(G)$	$= \Sigma_{as}(G) \cup \{$	$Acc(a)$ $Acc(d)$	$\}$
$\Sigma_s(G)$	$= \Sigma_{as}(G) \cup \{$	$Acc(a)$ $Acc(d)$ $\neg Acc(b) \rightarrow Acc(a)$ $\neg Acc(c) \rightarrow (Acc(b) \vee Acc(d))$	$\}$

Among the logical consequences of $\Sigma^0(G)$ *are the formulae:*

$\neg(Acc(a) \wedge Acc(b))$, $\neg(Acc(b) \wedge Acc(c))$ *and* $\neg(Acc(d) \wedge Acc(c))$

It follows that if a *s accepted then* b *is not accepted, and if* b *or* d *is accepted then* c *is not accepted.*

Among the logical consequences of $\Sigma_{as}(G)$ *are the formulae:*

$Val(\alpha)$, $Val(\beta)$ *and* $Val(\delta)$

$Act(\alpha) \leftrightarrow Gr(\alpha)$, $Act(\beta) \leftrightarrow Gr(\beta)$ *and* $Act(\delta) \leftrightarrow Gr(\delta)$

$Act(\alpha) \leftrightarrow Acc(a)$, $Act(\beta) \leftrightarrow Acc(b)$ *and* $Act(\delta) \leftrightarrow Acc(d)$

$\neg(Acc(a) \wedge Acc(b))$, $\neg(Acc(b) \wedge Acc(c))$ *and* $\neg(Acc(d) \wedge Acc(c))$

and so $Act(\alpha) \rightarrow \neg Acc(b)$, $Act(\beta) \rightarrow \neg Acc(c)$ *and* $Act(\delta) \rightarrow \neg Acc(c)$

It follows that α *(resp.* δ*) is active iff* a *(resp.* d*) is accepted,* β *is active iff* b *is accepted, if* α *is active then* b *is not accepted, and if* β *or* δ *is active then* c *is not accepted.*

Note that $\neg Acc(b)$ *and* $\neg Acc(c)$ *follow from* $\Sigma_r(G)$.

The following table displays the standard Dung extensions of $\mathcal{G}$*:*

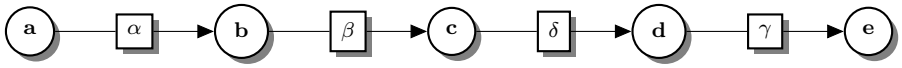
Semantics	*Extensions*
Grounded	$\{a, d\}$
Preferred	$\{a, d\}$
Stable	$\{a, d\}$
Complete	$\{a, d\}$

The following table shows the sets $S_{\mathcal{I}} = \{x \in \mathbf{A} | \mathcal{I}(Acc(x)) = \textit{true}\}$ *associated to the models* $\mathcal{I}$ *of some bases:*

Base	$S_{\mathcal{I}}$
$\Sigma(G)$	$\varnothing$ $\{a\}$ $\{b\}$ $\{c\}$ $\{d\}$ $\{a, d\}$ $\{b, d\}$ $\{a, c\}$
$\Sigma_d(G)$	$\varnothing$ $\{a\}$ $\{d\}$ $\{a, d\}$
$\Sigma_r(G)$	$\{a, d\}$

A.1.7 Example 7

Ex. 7



$\Sigma^0(G)$	$= \{$	$Acc(a) \rightarrow NAcc(b)$ $Acc(b) \rightarrow NAcc(c)$ $Acc(c) \rightarrow NAcc(d)$ $Acc(d) \rightarrow NAcc(e)$ $NAcc(b) \rightarrow \neg Acc(b)$ $NAcc(c) \rightarrow \neg Acc(c)$ $NAcc(d) \rightarrow \neg Acc(d)$ $NAcc(e) \rightarrow \neg Acc(e)$	$\}$
$\Sigma(G)$	$= \{$	$Gr(\alpha) \leftrightarrow Acc(a)$ $Gr(\beta) \leftrightarrow Acc(b)$ $Gr(\delta) \leftrightarrow Acc(c)$ $Gr(\gamma) \leftrightarrow Acc(d)$ $Act(\alpha) \leftrightarrow (Gr(\alpha) \wedge Val(\alpha))$ $Act(\beta) \leftrightarrow (Gr(\beta) \wedge Val(\beta))$ $Act(\delta) \leftrightarrow (Gr(\delta) \wedge Val(\delta))$ $Act(\gamma) \leftrightarrow (Gr(\gamma) \wedge Val(\gamma))$ $(Val(\alpha) \wedge Acc(a)) \rightarrow NAcc(b)$ $NAcc(b) \rightarrow \neg Acc(b)$ $(Val(\beta) \wedge Acc(b)) \rightarrow NAcc(c)$ $NAcc(c) \rightarrow \neg Acc(c)$ $(Val(\delta) \wedge Acc(c)) \rightarrow NAcc(d)$ $NAcc(d) \rightarrow \neg Acc(d)$ $(Val(\gamma) \wedge Acc(d)) \rightarrow NAcc(e)$ $NAcc(e) \rightarrow \neg Acc(e)$	$\}$
$\Sigma_{as}(G)$	$= \Sigma(G) \cup \{$	$Val(\alpha)$ $Val(\beta)$ $Val(\delta)$ $Val(\gamma)$	$\}$
$\Sigma_d(G)$	$= \Sigma_{as}(G) \cup \{$	$\neg Acc(b)$ $Acc(c) \rightarrow Acc(a)$ $Acc(d) \rightarrow Acc(b)$ $Acc(e) \rightarrow Acc(c)$	$\}$
$\Sigma_r(G)$	$= \Sigma(G)_{as} \cup \{$	$Acc(a)$ $Acc(a) \rightarrow Acc(c)$ $Acc(b) \rightarrow Acc(d)$ $Acc(c) \rightarrow Acc(e)$	$\}$
$\Sigma_s(G)$	$= \Sigma_{as}(G) \cup \{$	$Acc(a)$ $\neg Acc(b) \rightarrow Acc(a)$ $\neg Acc(c) \rightarrow Acc(b)$ $\neg Acc(d) \rightarrow Acc(c)$ $\neg Acc(e) \rightarrow Acc(d)$	$\}$

Among the logical consequences of $\Sigma^0(G)$ are the formulae:

$\neg(Acc(a) \wedge Acc(b))$, $\neg(Acc(b) \wedge Acc(c))$, $\neg(Acc(c) \wedge Acc(d))$ *and* $\neg(Acc(d) \wedge Acc(e))$

It follows that if a *(resp.* b*,* c*,* d*) is accepted, then* b *(resp.* c*,* d*,* e*) is not accepted.*

Among the logical consequences of $\Sigma_{as}(G)$ *are the formulae:*

$Val(\alpha)$, $Val(\beta)$, $Val(\delta)$ *and* $Val(\gamma)$

$Act(\alpha) \leftrightarrow Gr(\alpha)$, $Act(\beta) \leftrightarrow Gr(\beta)$, $Act(\delta) \leftrightarrow Gr(\delta)$ *and* $Act(\gamma) \leftrightarrow Gr(\gamma)$

$Act(\alpha) \leftrightarrow Acc(a)$, $Act(\beta) \leftrightarrow Acc(b)$, $Act(\delta) \leftrightarrow Acc(c)$ *and* $Act(\gamma) \leftrightarrow Acc(d)$

$\neg(Acc(a) \wedge Acc(b))$, $\neg(Acc(b) \wedge Acc(c))$, $\neg(Acc(c) \wedge Acc(d))$ *and*
$\neg(Acc(d) \wedge Acc(e))$

and so $Act(\alpha) \rightarrow \neg Acc(b)$, $Act(\beta) \rightarrow \neg Acc(c)$, $Act(\delta) \rightarrow \neg Acc(d)$ *and* $Act(\gamma) \rightarrow \neg Acc(e)$

It follows that α *(resp.* β*,* δ*,* γ*) is active iff* a *(resp.* b*,* c*,* d*) is accepted, and if* α *(resp.* β*,* δ*,* γ*) is active then* b *(resp.* c*,* d*,* e*) is not accepted.*

Note that:

$\neg Acc(d)$ *follows from* $\Sigma_d(G)$

$Acc(c)$, $Acc(e)$, $\neg Acc(b)$ *and* $\neg Acc(d)$ *follow from* $\Sigma_r(G)$.

The following table displays the standard Dung extensions of $\mathcal{G}$*:*

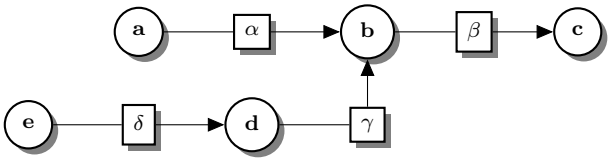
Semantics	*Extensions*
Grounded	$\{a, c, e\}$
Preferred	$\{a, c, e\}$
Stable	$\{a, c, e\}$
Complete	$\{a, c, e\}$

The following table shows the sets $S_{\mathcal{I}} = \{x \in \mathbf{A} | \mathcal{I}(Acc(x)) = \textit{true}\}$ *associated to the models* $\mathcal{I}$ *of some bases:*

Base	$S_{\mathcal{I}}$
$\Sigma(G)$	$\varnothing$ $\{a\}$ $\{b\}$ $\{c\}$ $\{d\}$ $\{e\}$ $\{a, c\}$ $\{a, d\}$ $\{a, e\}$ $\{b, d\}$ $\{b, e\}$ $\{c, e\}$ $\{a, c, e\}$
$\Sigma_d(G)$	$\varnothing$ $\{a\}$ $\{a, c\}$ $\{a, c, e\}$
$\Sigma_r(G)$	$\{a, c, e\}$

A.1.8 Example 8

Ex. 8



$\Sigma^0(G)$	$= \{$	$Acc(a) \rightarrow NAcc(b)$ $Acc(b) \rightarrow NAcc(c)$ $Acc(e) \rightarrow NAcc(d)$ $Acc(d) \rightarrow NAcc(b)$ $NAcc(b) \rightarrow \neg Acc(b)$ $NAcc(c) \rightarrow \neg Acc(c)$ $NAcc(d) \rightarrow \neg Acc(d)$	$\}$
$\Sigma(G)$	$= \{$	$Gr(\alpha) \leftrightarrow Acc(a)$ $Gr(\beta) \leftrightarrow Acc(b)$ $Gr(\delta) \leftrightarrow Acc(e)$ $Gr(\gamma) \leftrightarrow Acc(d)$ $Act(\alpha) \leftrightarrow (Gr(\alpha) \wedge Val(\alpha))$ $Act(\beta) \leftrightarrow (Gr(\beta) \wedge Val(\beta))$ $Act(\delta) \leftrightarrow (Gr(\delta) \wedge Val(\delta))$ $Act(\gamma) \leftrightarrow (Gr(\gamma) \wedge Val(\gamma))$ $(Val(\alpha) \wedge Acc(a)) \rightarrow NAcc(b)$ $NAcc(b) \rightarrow \neg Acc(b)$ $(Val(\beta) \wedge Acc(b)) \rightarrow NAcc(c)$ $NAcc(c) \rightarrow \neg Acc(c)$ $(Val(\delta) \wedge Acc(e)) \rightarrow NAcc(d)$ $NAcc(d) \rightarrow \neg Acc(d)$ $(Val(\gamma) \wedge Acc(d)) \rightarrow NAcc(b)$	$\}$
$\Sigma_{as}(G)$	$= \Sigma(G) \cup \{$	$Val(\alpha)$ $Val(\beta)$ $Val(\delta)$ $Val(\gamma)$	$\}$
$\Sigma_d(G)$	$= \Sigma_{as}(G) \cup \{$	$\neg Acc(d)$ $\neg Acc(b)$ $Acc(c) \rightarrow (Acc(a) \vee Acc(d))$	$\}$
$\Sigma_r(G)$	$= \Sigma_{as}(G) \cup \{$	$Acc(a)$ $Acc(e)$ $(Acc(a) \vee Acc(d)) \rightarrow Acc(c)$	$\}$
$\Sigma_s(G)$	$= \Sigma_{as}(G) \cup \{$	$Acc(a)$ $Acc(e)$ $\neg Acc(b) \rightarrow (Acc(a) \vee Acc(d))$ $\neg Acc(d) \rightarrow Acc(e)$ $\neg Acc(c) \rightarrow Acc(b)$	$\}$

Among the logical consequences of $\Sigma^0(G)$ *are the formulae:*

$\neg(Acc(a) \wedge Acc(b))$, $\neg(Acc(b) \wedge Acc(c))$, $\neg(Acc(e) \wedge Acc(d))$ *and* $\neg(Acc(d) \wedge Acc(b))$

It follows that if a (resp. b, e, d) is accepted then b (resp. c, d, b) is not accepted.

Among the logical consequences of $\Sigma_{as}(G)$ are the formulae:

$Val(\alpha)$, $Val(\beta)$, $Val(\delta)$ *and* $Val(\gamma)$

$Act(\alpha) \leftrightarrow Gr(\alpha)$, $Act(\beta) \leftrightarrow Gr(\beta)$, $Act(\delta) \leftrightarrow Gr(\delta)$ *and* $Act(\gamma) \leftrightarrow Gr(\gamma)$

$Act(\alpha) \leftrightarrow Acc(a)$, $Act(\beta) \leftrightarrow Acc(b)$, $Act(\delta) \leftrightarrow Acc(e)$ *and* $Act(\gamma) \leftrightarrow Acc(d)$

$\neg(Acc(a) \wedge Acc(b))$, $\neg(Acc(b) \wedge Acc(c))$, $\neg(Acc(e) \wedge Acc(d))$ *and* $\neg(Acc(d) \wedge Acc(b))$

and so $Act(\alpha) \rightarrow \neg Acc(b)$, $Act(\beta) \rightarrow \neg Acc(c)$, $Act(\delta) \rightarrow \neg Acc(d)$ *and* $Act(\gamma) \rightarrow \neg Acc(b)$

It follows that α (resp. β, δ, γ) is active iff a (resp. b, e, d) is accepted, if α or γ is active then b is not accepted and if β (resp. δ) is active then c (resp. d) is not accepted.

Note that $Acc(c)$, $\neg Acc(b)$ and $\neg Acc(d)$ follow from $\Sigma_r(G)$.

The following table displays the standard Dung extensions of $\mathcal{G}$:

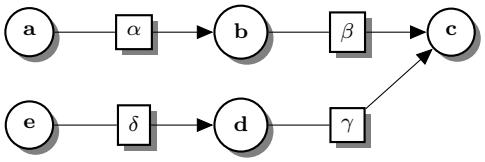
Semantics	*Extensions*
Grounded	$\{a, c, e\}$
Preferred	$\{a, c, e\}$
Stable	$\{a, c, e\}$
Complete	$\{a, c, e\}$

The following table shows the sets $S_{\mathcal{I}} = \{x \in \mathbf{A} | \mathcal{I}(Acc(x)) = true\}$ associated to the models $\mathcal{I}$ of some bases:

Base	$S_{\mathcal{I}}$
$\Sigma(G)$	$\varnothing$ $\{a\}$ $\{b\}$ $\{c\}$ $\{d\}$ $\{e\}$ $\{a, c\}$ $\{a, d\}$ $\{a, e\}$ $\{b, e\}$ $\{c, e\}$ $\{c, d\}$ $\{a, c, e\}$
$\Sigma_d(G)$	$\varnothing$ $\{a\}$ $\{e\}$ $\{a, c\}$ $\{a, e\}$ $\{a, c, e\}$
$\Sigma_r(G)$	$\{a, c, e\}$

A.1.9 Example 9

Ex. 9



$\Sigma^0(G)$	$= \{$	$Acc(a) \rightarrow NAcc(b)$ $Acc(b) \rightarrow NAcc(c)$ $Acc(e) \rightarrow NAcc(d)$ $Acc(d) \rightarrow NAcc(c)$ $NAcc(b) \rightarrow \neg Acc(b)$ $NAcc(c) \rightarrow \neg Acc(c)$ $NAcc(d) \rightarrow \neg Acc(d)$	$\}$
$\Sigma(G)$	$= \{$	$Gr(\alpha) \leftrightarrow Acc(a)$ $Gr(\beta) \leftrightarrow Acc(b)$ $Gr(\delta) \leftrightarrow Acc(e)$ $Gr(\gamma) \leftrightarrow Acc(d)$ $Act(\alpha) \leftrightarrow (Gr(\alpha) \wedge Val(\alpha))$ $Act(\beta) \leftrightarrow (Gr(\beta) \wedge Val(\beta))$ $Act(\delta) \leftrightarrow (Gr(\delta) \wedge Val(\delta))$ $Act(\gamma) \leftrightarrow (Gr(\gamma) \wedge Val(\gamma))$ $(Val(\alpha) \wedge Acc(a)) \rightarrow NAcc(b)$ $NAcc(b) \rightarrow \neg Acc(b)$ $(Val(\beta) \wedge Acc(b)) \rightarrow NAcc(c)$ $NAcc(c) \rightarrow \neg Acc(c)$ $(Val(\delta) \wedge Acc(e)) \rightarrow NAcc(d)$ $NAcc(d) \rightarrow \neg Acc(d)$ $(Val(\gamma) \wedge Acc(d)) \rightarrow NAcc(c)$	$\}$
$\Sigma_{as}(G)$	$= \Sigma(G) \cup \{$	$Val(\alpha)$ $Val(\beta)$ $Val(\delta)$ $Val(\gamma)$	$\}$
$\Sigma_d(G)$	$= \Sigma_{as}(G) \cup \{$	$\neg Acc(d)$ $\neg Acc(b)$ $Acc(c) \rightarrow (Acc(a) \wedge Acc(e))$	$\}$
$\Sigma_r(G)$	$= \Sigma_{as}(G) \cup \{$	$Acc(a)$ $Acc(e)$ $(Acc(a) \wedge Acc(e)) \rightarrow Acc(c)$	$\}$
$\Sigma_s(G)$	$= \Sigma_{as}(G) \cup \{$	$Acc(a)$ $Acc(e)$ $\neg Acc(b) \rightarrow Acc(a)$ $\neg Acc(c) \rightarrow (Acc(b) \vee Acc(d))$ $\neg Acc(d) \rightarrow Acc(e)$	$\}$

Among the logical consequences of $\Sigma^0(G)$ *are the formulae:*

$\neg(Acc(a) \wedge Acc(b))$, $\neg(Acc(b) \wedge Acc(c))$, $\neg(Acc(e) \wedge Acc(d))$ *and* $\neg(Acc(d) \wedge Acc(c))$

It follows that if a *(resp.* b*,* e*,* d*) is accepted,* b *(resp.* c*,* d*,* c*) is not accepted.*

Among the logical consequences of $\Sigma_{as}(G)$ *are the formulae:*

$Val(\alpha)$, $Val(\beta)$, $Val(\delta)$ *and* $Val(\gamma)$

$Act(\alpha) \leftrightarrow Gr(\alpha)$, $Act(\beta) \leftrightarrow Gr(\beta)$, $Act(\delta) \leftrightarrow Gr(\delta)$ *and* $Act(\gamma) \leftrightarrow Gr(\gamma)$

$Act(\alpha) \leftrightarrow Acc(a)$, $Act(\beta) \leftrightarrow Acc(b)$, $Act(\delta) \leftrightarrow Acc(e)$ *and* $Act(\gamma) \leftrightarrow Acc(d)$

$\neg(Acc(a) \wedge Acc(b))$, $\neg(Acc(b) \wedge Acc(c))$, $\neg(Acc(e) \wedge Acc(d))$ *and* $\neg(Acc(d) \wedge Acc(c))$

and so $Act(\alpha) \rightarrow \neg Acc(b)$, $Act(\beta) \rightarrow \neg Acc(c)$, $Act(\delta) \rightarrow \neg Acc(d)$ *and* $Act(\gamma) \rightarrow \neg Acc(c)$

It follows that α *(resp.* β*,* δ*,* γ*) is active iff* a *(resp.* b*,* e*,* d*) is accepted, if* β *or* γ *is active then* c *is not accepted and if* α *(resp.* δ*) is active then* b *(resp.* d*) is not accepted.*

Note that $Acc(c)$, $\neg Acc(b)$ *and* $\neg Acc(d)$ *follow from* $\Sigma_r(G)$.

The following table displays the standard Dung extensions of $\mathcal{G}$*:*

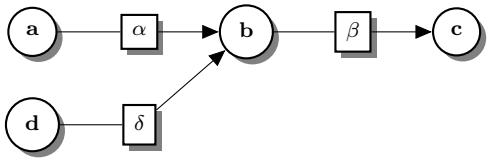
Semantics	*Extensions*
Grounded	$\{a, c, e\}$
Preferred	$\{a, c, e\}$
Stable	$\{a, c, e\}$
Complete	$\{a, c, e\}$

The following table shows the sets $S_{\mathcal{I}} = \{x \in \mathbf{A} | \mathcal{I}(Acc(x)) = true\}$ *associated to the models* $\mathcal{I}$ *of some bases:*

Base	$S_{\mathcal{I}}$
$\Sigma(G)$	$\varnothing$ $\{a\}$ $\{b\}$ $\{c\}$ $\{d\}$ $\{e\}$ $\{a, c\}$ $\{a, d\}$ $\{a, e\}$ $\{b, e\}$ $\{c, e\}$ $\{b, d\}$ $\{a, c, e\}$
$\Sigma_d(G)$	$\varnothing$ $\{a\}$ $\{e\}$ $\{a, e\}$ $\{a, c, e\}$
$\Sigma_r(G)$	$\{a, c, e\}$

A.1.10 Example 10

Ex. 10



$\Sigma^0(G)$	$= \{$	$Acc(a) \rightarrow NAcc(b)$ $Acc(b) \rightarrow NAcc(c)$ $Acc(d) \rightarrow NAcc(b)$ $NAcc(b) \rightarrow \neg Acc(b)$ $NAcc(c) \rightarrow \neg Acc(c)$	$\}$
$\Sigma(G)$	$= \{$	$Gr(\alpha) \leftrightarrow Acc(a)$ $Gr(\beta) \leftrightarrow Acc(b)$ $Gr(\delta) \leftrightarrow Acc(d)$ $Act(\alpha) \leftrightarrow (Gr(\alpha) \wedge Val(\alpha))$ $Act(\beta) \leftrightarrow (Gr(\beta) \wedge Val(\beta))$ $Act(\delta) \leftrightarrow (Gr(\delta) \wedge Val(\delta))$ $(Val(\alpha) \wedge Acc(a)) \rightarrow NAcc(b)$ $NAcc(b) \rightarrow \neg Acc(b)$ $(Val(\beta) \wedge Acc(b)) \rightarrow NAcc(c)$ $NAcc(c) \rightarrow \neg Acc(c)$ $(Val(\delta) \wedge Acc(d)) \rightarrow NAcc(b)$	$\}$
$\Sigma_{as}(G)$	$= \Sigma(G) \cup \{$	$Val(\alpha)$ $Val(\beta)$ $Val(\delta)$	$\}$
$\Sigma_d(G)$	$= \Sigma_{as}(G) \cup \{$	$\neg Acc(b)$ $Acc(c) \rightarrow (Acc(a) \vee Acc(d))$	$\}$
$\Sigma_r(G)$	$= \Sigma_{as}(G) \cup \{$	$Acc(a)$ $Acc(d)$ $(Acc(a) \vee Acc(d)) \rightarrow Acc(c)$	$\}$
$\Sigma_s(G)$	$= \Sigma_{as}(G) \cup \{$	$Acc(a)$ $Acc(d)$ $\neg Acc(b) \rightarrow (Acc(a) \vee Acc(d))$ $\neg Acc(c) \rightarrow Acc(b)$	$\}$

Among the logical consequences of $\Sigma^0(G)$ *are the formulae:*

$\neg(Acc(a) \wedge Acc(b))$, $\neg(Acc(b) \wedge Acc(c))$ *and* $\neg(Acc(d) \wedge Acc(b))$

It follows that if a *(resp.* b*,* d*) is accepted,* b *(resp.* c*,* b*) is not accepted.*

Among the logical consequences of $\Sigma_{as}(G)$ *are the formulae:*

$Val(\alpha)$, $Val(\beta)$ *and* $Val(\delta)$

$Act(\alpha) \leftrightarrow Gr(\alpha)$, $Act(\beta) \leftrightarrow Gr(\beta)$ *and* $Act(\delta) \leftrightarrow Gr(\delta)$

$Act(\alpha) \leftrightarrow Acc(a)$, $Act(\beta) \leftrightarrow Acc(b)$ *and* $Act(\delta) \leftrightarrow Acc(d)$

$\neg(Acc(a) \wedge Acc(b))$, $\neg(Acc(b) \wedge Acc(c))$ *and* $\neg(Acc(d) \wedge Acc(b))$

and so $Act(\alpha) \rightarrow \neg Acc(b)$, $Act(\beta) \rightarrow \neg Acc(c)$ *and* $Act(\delta) \rightarrow \neg Acc(b)$

It follows that α *(resp.* β*,* δ*) is active iff* a *(resp.* b*,* d*) is accepted, if* β *is active then* c *is not accepted and if* α *or* δ *is active then* b *is not accepted.*

Note that $\neg Acc(b)$ *and* $Acc(c)$ *follow from* $\Sigma_r(G)$*.*

The following table displays the standard Dung extensions of $\mathcal{G}$*:*

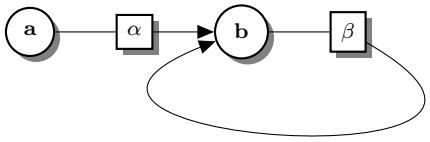
Semantics	*Extensions*
Grounded	$\{a, c, d\}$
Preferred	$\{a, c, d\}$
Stable	$\{a, c, d\}$
Complete	$\{a, c, d\}$

The following table shows the sets $S_{\mathcal{I}} = \{x \in \mathbf{A} | \mathcal{I}(Acc(x)) = true\}$ *associated to the models* $\mathcal{I}$ *of some bases:*

Base	$S_{\mathcal{I}}$
$\Sigma(G)$	$\varnothing$ $\{a\}$ $\{b\}$ $\{c\}$ $\{d\}$ $\{a, c\}$ $\{a, d\}$ $\{c, d\}$ $\{a, c, d\}$
$\Sigma_d(G)$	$\varnothing$ $\{a\}$ $\{d\}$ $\{a, c\}$ $\{a, d\}$ $\{c, d\}$ $\{a, c, d\}$
$\Sigma_r(G)$	$\{a, c, d\}$

A.1.11 Example 11

Ex. 11



$\Sigma^0(G)$	$= \{$	$Acc(a) \rightarrow NAcc(b)$ $NAcc(b) \rightarrow \neg Acc(b)$ $Acc(b) \rightarrow NAcc(b)$	$\}$
$\Sigma(G)$	$= \{$	$Gr(\alpha) \leftrightarrow Acc(a)$ $Gr(\beta) \leftrightarrow Acc(b)$ $Act(\alpha) \leftrightarrow (Gr(\alpha) \wedge Val(\alpha))$ $Act(\beta) \leftrightarrow (Gr(\beta) \wedge Val(\beta))$ $(Val(\alpha) \wedge Acc(a)) \rightarrow NAcc(b)$ *et* $NAcc(b) \rightarrow \neg Acc(b)$ $(Val(\beta) \wedge Acc(b)) \rightarrow NAcc(b)$ *et* $NAcc(b) \rightarrow \neg Acc(b)$	$\}$
$\Sigma_{as}(G)$	$= \Sigma(G) \cup \{$	$Val(\alpha)$ $Val(\beta)$	$\}$
$\Sigma_d(G)$	$= \Sigma_{as}(G) \cup \{$	$\neg Acc(b)$	$\}$
$\Sigma_r(G)$	$= \Sigma_{as}(G) \cup \{$	$Acc(a)$	$\}$
$\Sigma_s(G)$	$= \Sigma_{as}(G) \cup \{$	$Acc(a)$ $\rightarrow Acc(b), Acc(a)$	$\}$

The formula $\neg Acc(b)$ *is a logical consequence of* $\Sigma^0(G)$.

Among the logical consequences of $\Sigma_{as}(G)$ *are the formulae:*

$Val(\alpha)$ *and* $Val(\beta)$

$Act(\alpha) \leftrightarrow Gr(\alpha)$ *and* $Act(\beta) \leftrightarrow Gr(\beta)$ *and so* $Act(\alpha) \leftrightarrow Acc(a)$, *and* $Act(\beta) \leftrightarrow Acc(b)$

$\neg(Acc(a) \wedge Acc(b))$ *and so* $Act(\alpha) \rightarrow \neg Acc(b)$, *and so* $Act(\beta) \rightarrow \neg Acc(b)$

La formule $\neg Acc(b)$ *is also a logical consequence of* $\Sigma(G)$ *and of* $\Sigma_r(G)$.

Note that the property $\mathbf{P}_{\text{reins}}$ *applied to* b *produces a tautology:* $((Acc(a) \vee Acc(b)) \wedge \bot) \rightarrow Acc(b)$. *Indeed,* b *must be defended both against* a *and against itself. Obviously,* a *(or* b*) can be used to defend* b *against itself. However, no argument enables to defend* b *against* a.

The following table displays the standard Dung extensions of $\mathcal{G}$*:*

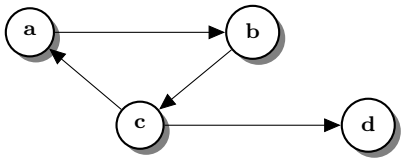
Semantics	*Extensions*
Grounded	$\{a\}$
Preferred	$\{a\}$
Stable	$\{a\}$
Complete	$\{a\}$

The following table shows the sets $S_{\mathcal{I}} = \{x \in \mathbf{A} | \mathcal{I}(Acc(x)) = \textit{true}\}$ *associated to the models* $\mathcal{I}$ *of some bases:*

Base	$S_{\mathcal{I}}$
$\Sigma^0(G)$	$\varnothing$ $\{a\}$
$\Sigma_d(G)$	$\varnothing$ $\{a\}$
$\Sigma_r(G)$	$\{a\}$

A.1.12 Example 12

Ex. 12 *In this example we do not use the label of interactions. So here we only give the simplified form of bases.*



$\Sigma^0(G)$	$= \{$	$Acc(a) \rightarrow NAcc(b)$ $Acc(b) \rightarrow NAcc(c)$ $Acc(c) \rightarrow NAcc(a)$ $Acc(c) \rightarrow NAcc(d)$ $NAcc(a) \rightarrow \neg Acc(a)$ $NAcc(b) \rightarrow \neg Acc(b)$ $NAcc(c) \rightarrow \neg Acc(c)$ $NAcc(d) \rightarrow \neg Acc(d)$	$\}$
$\Sigma^0_d(G)$	$= \Sigma^0(G) \cup \{$	$Acc(a) \rightarrow Acc(b)$ $Acc(b) \rightarrow Acc(c)$ $Acc(c) \rightarrow Acc(a)$ $Acc(d) \rightarrow Acc(b)$	$\}$
$\Sigma^0_r(G)$	$= \Sigma^0(G) \cup \{$	$Acc(a) \rightarrow Acc(c)$ $Acc(b) \rightarrow Acc(a)$ $Acc(c) \rightarrow Acc(b)$ $Acc(b) \rightarrow Acc(d)$	$\}$
$\Sigma^0_s(G)$	$= \Sigma^0(G) \cup \{$	$\neg Acc(a) \rightarrow Acc(c)$ $\neg Acc(b) \rightarrow Acc(a)$ $\neg Acc(c) \rightarrow Acc(b)$ $\neg Acc(d) \rightarrow Acc(c)$	$\}$

The following table displays the standard Dung extensions of $\mathcal{G}$:

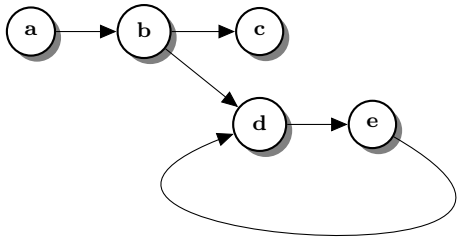
Semantics	*Extensions*
Grounded	$\varnothing$
Preferred	$\varnothing$
Stable	*none*
Complete	$\varnothing$

The following table shows the sets $S_{\mathcal{I}} = \{x \in \mathbf{A} | \mathcal{I}(Acc(x)) = true\}$ associated to the models $\mathcal{I}$ of some bases:

Base	$S_{\mathcal{I}}$
$\Sigma^0(G)$	$\varnothing$ $\{a\}$ $\{b\}$ $\{c\}$ $\{d\}$ $\{a,d\}$ $\{b,d\}$
$\Sigma_d(G)$	$\varnothing$
$\Sigma_r(G)$	$\varnothing$

A.1.13 Example 13

Ex. 13 *In this example we do not use the label of interactions. So here we only give the simplified form of bases.*



$\Sigma^0(G)$	$= \{$	$Acc(a) \rightarrow NAcc(b)$ $Acc(b) \rightarrow NAcc(c)$ $Acc(b) \rightarrow NAcc(d)$ $Acc(d) \rightarrow NAcc(e)$ $Acc(e) \rightarrow NAcc(d)$ $NAcc(b) \rightarrow \neg Acc(b)$ $NAcc(c) \rightarrow \neg Acc(c)$ $NAcc(d) \rightarrow \neg Acc(d)$ $NAcc(e) \rightarrow \neg Acc(e)$
$\Sigma^0_d(G)$	$= \Sigma^0(G) \cup \{$	$Acc(c) \rightarrow Acc(a)$ $Acc(d) \rightarrow (Acc(a) \wedge Acc(d))$ $Acc(e) \rightarrow (Acc(b) \vee Acc(e))$ $\neg Acc(b)$
$\Sigma^0_r(G)$	$= \Sigma^0(G) \cup \{$	$Acc(a) \rightarrow Acc(c)$ $(Acc(a) \wedge Acc(d)) \rightarrow Acc(d)$ $(Acc(b) \vee Acc(e)) \rightarrow Acc(e)$ $Acc(a)$
$\Sigma^0_s(G)$	$= \Sigma^0(G) \cup \{$	$Acc(a)$ $\neg Acc(b) \rightarrow Acc(a)$ $\neg Acc(c) \rightarrow Acc(b)$ $\neg Acc(d) \rightarrow (Acc(b) \vee Acc(e))$ $\neg Acc(e) \rightarrow Acc(d)$

The following table displays the standard Dung extensions of $\mathcal{G}$:

Semantics	*Extensions*
Grounded	$\{a, c\}$
Preferred	$\{a, c, d\}$, $\{a, c, e\}$
Stable	$\{a, c, d\}$, $\{a, c, e\}$
Complete	$\{a, c\}$, $\{a, c, d\}$, $\{a, c, e\}$

The following table shows the sets $S_{\mathcal{I}} = \{x \in \mathbf{A} | \mathcal{I}(Acc(x)) = \textit{true}\}$ associated to the models $\mathcal{I}$ of some bases:

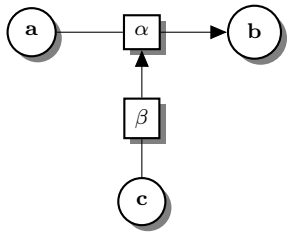
Base	$S_{\mathcal{I}}$
$\Sigma^0(G)$	$\varnothing$ $\{a\}$ $\{b\}$ $\{c\}$ $\{d\}$ $\{e\}$ $\{a,c\}$ $\{a,d\}$ $\{a,e\}$ $\{c,e\}$ $\{c,d\}$ $\{b,e\}$ $\{a,c,d\}$ $\{a,c,e\}$
$\Sigma_d(G)$	$\varnothing$ $\{a\}$ $\{e\}$ $\{a,c\}$ $\{a,d\}$ $\{a,e\}$ $\{a,c,d\}$ $\{a,c,e\}$
$\Sigma_r(G)$	$\{a,c\}$ $\{a,c,d\}$ $\{a,c,e\}$

A.2 Examples with recursivity (ASAF)

In the following examples, the bases $\Sigma^0(G)$ and $\Sigma_{as}(G)$ are not pertinent.

A.2.1 Example 14

Ex. 14

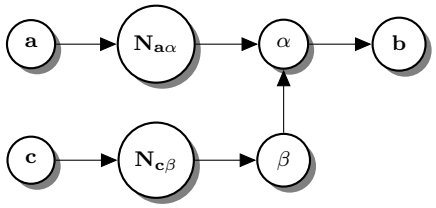


$\Sigma(G)$	$= \{$	$Gr(\alpha) \leftrightarrow Acc(a)$ $Gr(\beta) \leftrightarrow Acc(c)$ $(Val(\beta) \wedge Acc(c)) \rightarrow \neg Val(\alpha)$ $Act(\alpha) \leftrightarrow (Gr(\alpha) \wedge Val(\alpha))$ $Act(\beta) \leftrightarrow (Gr(\beta) \wedge Val(\beta))$ $(Val(\alpha) \wedge Acc(a)) \rightarrow NAcc(b)$ $NAcc(b) \rightarrow \neg Acc(b)$	$\}$
$\Sigma_d(G)$	$= \Sigma(G) \cup \{$	$Acc(b) \rightarrow (Val(\beta) \wedge Acc(c))$ $\neg Val(\alpha)$	$\}$
$\Sigma_r(G)$	$= \Sigma(G) \cup \{$	$Acc(a)$ $Acc(c)$ $(Val(\beta) \wedge Acc(c)) \rightarrow Acc(b)$ $Val(\beta)$	$\}$
$\Sigma_s(G)$	$= \Sigma(G) \cup \{$	$Acc(a)$ $\neg Acc(b) \rightarrow (Val(\alpha \wedge Acc(a))$ $Acc(c)$ $\neg Val(\alpha) \rightarrow (Val(\beta) \wedge Acc(c))$ $Val(\beta)$	$\}$

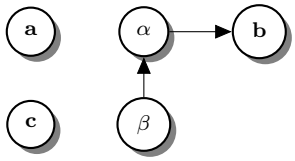
Using definition of Section 6.2 on page 54, some specific structures of this graph are displayed in the following table:

Semantics	*Structures*
Admissible	$(\varnothing, \{\beta\})$, $(\{a\}, \{\beta\})$, $(\{c\}, \{\beta\})$,$(\{a, c\}, \{\beta\})$, $(\{b, c\}, \{\beta\})$, $(\{a, b, c\}, \{\beta\})$
Stable	$(\{a, b, c\}, \{\beta\})$
Complete	$(\{a, b, c\}, \{\beta\})$

In the case when recursive attacks are encoded with meta-arguments, as in [15], the graph is turned into:



With [6, 21], the graph is turned into:

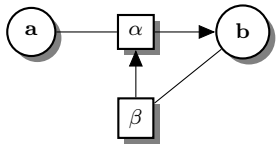


In both cases, the standard Dung extensions of these new graphs are displayed in the following table:

Semantics	*Extensions*
Grounded	$\{a, b, c, \beta\}$
Preferred	$\{a, b, c, \beta\}$
Stable	$\{a, b, c, \beta\}$
Complete	$\{a, b, c, \beta\}$

A.2.2 Example 15

Ex. 15

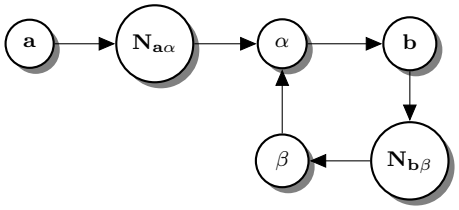


$\Sigma(G)$	$= \{$	$Gr(\alpha) \leftrightarrow Acc(a)$ $Gr(\beta) \leftrightarrow Acc(b)$ $(Val(\beta) \wedge Acc(b)) \rightarrow \neg Val(\alpha)$ $Act(\alpha) \leftrightarrow (Gr(\alpha) \wedge Val(\alpha))$ $Act(\beta) \leftrightarrow (Gr(\beta) \wedge Val(\beta))$ $(Val(\alpha) \wedge Acc(a)) \rightarrow NAcc(b)$ $NAcc(b) \rightarrow \neg Acc(b)$	$\}$
$\Sigma_d(G)$	$= \Sigma(G) \cup \{$	$Acc(b) \rightarrow Val(\beta)$ $Val(\alpha) \rightarrow Acc(a)$	$\}$
$\Sigma_r(G)$	$= \Sigma(G) \cup \{$	$Acc(a)$ $Val(\beta)$	$\}$
$\Sigma_s(G)$	$= \Sigma(G) \cup \{$	$Acc(a)$ $\neg Acc(b) \rightarrow (Val(\alpha) \wedge Acc(a))$ $Val(\beta)$ $\neg Val(\alpha) \rightarrow (Val(\beta) \wedge Acc(b))$	$\}$

Using definition of Section 6.2 on page 54, some specific structures of this graph are displayed in the following table:

Semantics	*Structures*
Admissible	$(\varnothing, \{\beta\})$, $(\{a\}, \{\beta\})$, $(\{b\}, \{\beta\})$, $(\{a\}, \{\alpha, \beta\})$, $(\{a, b\}, \{\beta\})$
Stable	$(\{a\}, \{\alpha, \beta\})$, $(\{a, b\}, \{\beta\})$
Complete	$(\{a\}, \{\beta\})$, $(\{a\}, \{\alpha, \beta\})$, $(\{a, b\}, \{\beta\})$

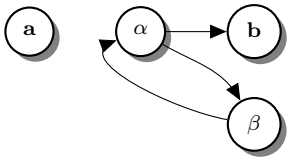
In the case when recursive attacks are encoded with meta-arguments, as in [15], the graph is turned into:



The standard Dung extensions of this new graph are displayed in the following table:

Semantics	*Extensions*
Grounded	$\{a\}$
Preferred	$\{a, b, \beta\}$ $\{a, \alpha, N_{b\beta}\}$
Stable	$\{a, b, \beta\}$ $\{a, \alpha, N_{b\beta}\}$
Complete	$\{a\}$ $\{a, b, \beta\}$ $\{a, \alpha, N_{b\beta}\}$

With [6, 21], the graph is turned into:

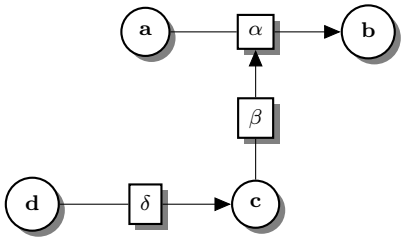


The standard Dung extensions of this new graph are displayed in the following table (they are very similar to the ones obtained with MAS approach):

Semantics	*Extensions*
Grounded	$\{a\}$
Preferred	$\{a, b, \beta\}$ $\{a, \alpha\}$
Stable	$\{a, b, \beta\}$ $\{a, \alpha\}$
Complete	$\{a\}$ $\{a, b, \beta\}$ $\{a, \alpha\}$

A.2.3 Example 16

Ex. 16

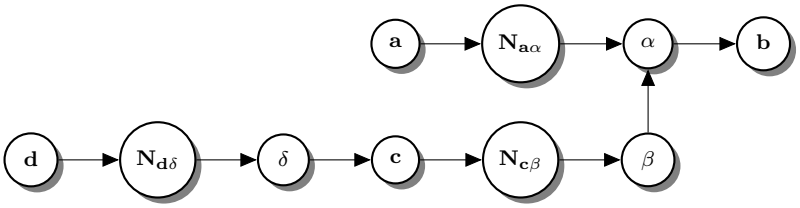


$\Sigma(G)$	$= \{$	$Gr(\alpha) \leftrightarrow Acc(a)$ $Gr(\beta) \leftrightarrow Acc(c)$ $Gr(\delta) \leftrightarrow Acc(d)$ $(Val(\beta) \wedge Acc(c)) \rightarrow \neg Val(\alpha)$ $Act(\alpha) \leftrightarrow (Gr(\alpha) \wedge Val(\alpha))$ $Act(\beta) \leftrightarrow (Gr(\beta) \wedge Val(\beta))$ $Act(\delta) \leftrightarrow (Gr(\delta) \wedge Val(\delta))$ $(Val(\alpha) \wedge Acc(a)) \rightarrow NAcc(b)$ $NAcc(b) \rightarrow \neg Acc(b)$ $(Val(\delta) \wedge Acc(d)) \rightarrow NAcc(c)$ $NAcc(c) \rightarrow \neg Acc(c)$	$\}$
$\Sigma_d(G)$	$= \Sigma(G) \cup \{$	$Acc(b) \rightarrow (Val(\beta) \wedge Acc(c))$ $\neg Acc(c)$ $Val(\alpha) \rightarrow (Val(\delta) \wedge Acc(d))$	$\}$
$\Sigma_r(G)$	$= \Sigma(G) \cup \{$	$Acc(a)$ $Acc(d)$ $(Val(\beta) \wedge Acc(c)) \rightarrow Acc(b)$ $Val(\beta)$ $Val(\delta)$ $(Val(\delta) \wedge Acc(d)) \rightarrow Val(\alpha)$	$\}$
$\Sigma_s(G)$	$= \Sigma(G) \cup \{$	$Acc(a)$ $Acc(d)$ $\neg Acc(c) \rightarrow (Val(\delta) \wedge Acc(d))$ $\neg Acc(b) \rightarrow (Val(\alpha) \wedge Acc(a))$ $Val(\beta)$ $Val(\delta)$ $\neg Val(\alpha) \rightarrow (Val(\beta) \wedge Acc(c))$	$\}$

Using definition of Section 6.2 on page 54, some specific structures of this graph are displayed in the following table:

Semantics	*Structures*
Admissible	$(\varnothing, \{\beta, \delta\})$, $(\{a\}, \{\beta, \delta\})$, $(\{d\}, \{\beta, \delta\})$, $(\{d\}, \{\alpha, \beta, \delta\})$, $(\{a, d\}, \{\beta, \delta\})$, $(\{a, d\}, \{\alpha, \beta, \delta\})$
Stable	$(\{a, d\}, \{\alpha, \beta, \delta\})$
Complete	$(\{a, d\}, \{\alpha, \beta, \delta\})$

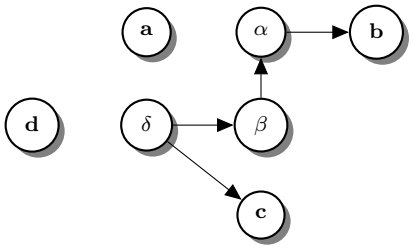
In the case when recursive attacks are encoded with meta-arguments, as in [15], the graph is turned into (note that as δ is not attacked, it can be handled as a simple attack) :



The standard Dung extensions of this new graph are displayed in the following table:

Semantics	*Extensions*
Grounded	$\{a, d, N_{c\beta}, \alpha, \delta\}$
Preferred	$\{a, d, N_{c\beta}, \alpha, \delta\}$
Stable	$\{a, d, N_{c\beta}, \alpha, \delta\}$
Complete	$\{a, d, N_{c\beta}, \alpha, \delta\}$

With [6, 21], the graph is turned into:

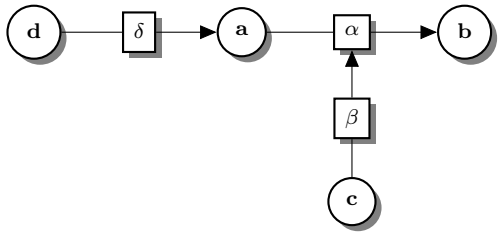


The standard Dung extensions of this new graph are displayed in the following table (similar to those obtained with MAS approach):

Semantics	*Extensions*
Grounded	$\{a, d, \alpha, \delta\}$
Preferred	$\{a, d, \alpha, \delta\}$
Stable	$\{a, d, \alpha, \delta\}$
Complete	$\{a, d, \alpha, \delta\}$

A.2.4 Example 17

Ex. 17

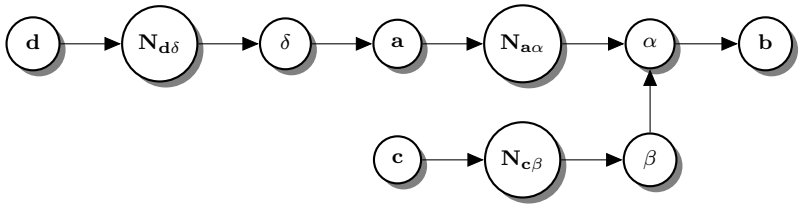


$\Sigma(G)$	$= \{$	$Gr(\alpha) \leftrightarrow Acc(a)$ $Gr(\beta) \leftrightarrow Acc(c)$ $Gr(\delta) \leftrightarrow Acc(d)$ $(Val(\beta) \wedge Acc(c)) \rightarrow \neg Val(\alpha)$ $Act(\alpha) \leftrightarrow (Gr(\alpha) \wedge Val(\alpha))$ $Act(\beta) \leftrightarrow (Gr(\beta) \wedge Val(\beta))$ $Act(\delta) \leftrightarrow (Gr(\delta) \wedge Val(\delta))$ $(Val(\alpha) \wedge Acc(a)) \rightarrow NAcc(b)$ $NAcc(b) \rightarrow \neg Acc(b)$ $(Val(\delta) \wedge Acc(d)) \rightarrow NAcc(a)$ $NAcc(a) \rightarrow \neg Acc(a)$	$\}$
$\Sigma_d(G)$	$= \Sigma(G) \cup \{$	$\neg Acc(a)$ $Acc(b) \rightarrow ((Val(\beta) \wedge Acc(c)) \vee (Val(\delta) \wedge Acc(d)))$ $\neg Val(\alpha)$	$\}$
$\Sigma_r(G)$	$= \Sigma(G) \cup \{$	$Acc(d)$ $Acc(c)$ $((Val(\beta) \wedge Acc(c)) \vee (Val(\delta) \wedge Acc(d))) \rightarrow Acc(b)$ $Val(\beta)$ $Val(\delta)$	$\}$
$\Sigma_s(G)$	$= \Sigma(G) \cup \{$	$Acc(d)$ $Acc(c)$ $\neg Acc(a) \rightarrow (Val(\delta) \wedge Acc(d))$ $\neg Acc(b) \rightarrow (Val(\alpha) \wedge Acc(a))$ $Val(\beta)$ $Val(\delta)$ $\neg Val(\alpha) \rightarrow (Val(\beta) \wedge Acc(c))$	$\}$

Using definition of Section 6.2 on page 54, some specific structures of this graph are displayed in the following table:

Semantics	*Structures*
Admissible	$(\varnothing, \{\beta, \delta\})$, $(\{c\}, \{\beta, \delta\})$, $(\{d\}, \{\beta, \delta\})$, $(\{b, c\}, \{\beta, \delta\})$, $(\{b, d\}, \{\beta, \delta\})$, $(\{c, d\}, \{\beta, \delta\})$, $(\{b, c, d\}, \{\beta, \delta\})$
Stable	$(\{b, c, d\}, \{\beta, \delta\})$
Complete	$(\{b, c, d\}, \{\beta, \delta\})$

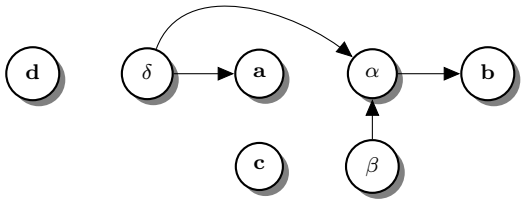
In the case when recursive attacks are encoded with meta-arguments, as in [15], the graph is turned into (note that as δ is not attacked, it can be handled as a simple attack):



The standard Dung extensions of this new graph are displayed in the following table:

Semantics	*Extensions*
Grounded	$\{c, d, \delta, N_{a\alpha}, \beta, b\}$
Preferred	$\{c, d, \delta, N_{a\alpha}, \beta, b\}$
Stable	$\{c, d, \delta, N_{a\alpha}, \beta, b\}$
Complete	$\{c, d, \delta, N_{a\alpha}, \beta, b\}$

With [6, 21], the graph is turned into:

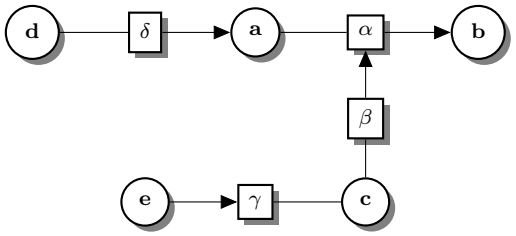


The standard Dung extensions of this new graph are displayed in the following table (similar to those obtained with MAS approach):

Semantics	*Extensions*
Grounded	$\{c, d, \delta, \beta, b\}$
Preferred	$\{c, d, \delta, \beta, b\}$
Stable	$\{c, d, \delta, \beta, b\}$
Complete	$\{c, d, \delta, \beta, b\}$

A.2.5 Example 18

Ex. 18

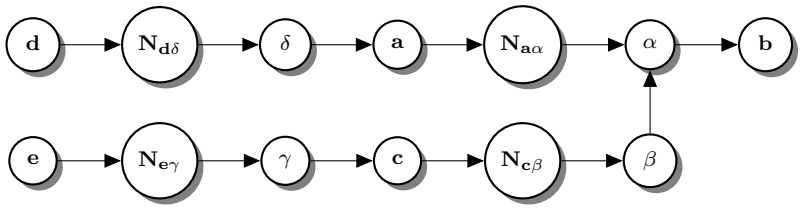


$\Sigma(G)$	$= \{$	$Gr(\alpha) \leftrightarrow Acc(a)$ $Gr(\beta) \leftrightarrow Acc(c)$ $Gr(\delta) \leftrightarrow Acc(d)$ $Gr(\gamma) \leftrightarrow Acc(e)$ $(Val(\beta) \wedge Acc(c)) \rightarrow \neg Val(\alpha)$ $Act(\alpha) \leftrightarrow (Gr(\alpha) \wedge Val(\alpha))$ $Act(\beta) \leftrightarrow (Gr(\beta) \wedge Val(\beta))$ $Act(\delta) \leftrightarrow (Gr(\delta) \wedge Val(\delta))$ $Act(\gamma) \leftrightarrow (Gr(\gamma) \wedge Val(\gamma))$ $(Val(\alpha) \wedge Acc(a)) \rightarrow NAcc(b)$ $NAcc(b) \rightarrow \neg Acc(b)$ $(Val(\delta) \wedge Acc(d)) \rightarrow NAcc(a)$ $NAcc(a) \rightarrow \neg Acc(a)$ $(Val(\gamma) \wedge Acc(e)) \rightarrow NAcc(c)$ $NAcc(c) \rightarrow \neg Acc(c)$	$\}$
$\Sigma_d(G)$	$= \Sigma(G) \cup \{$	$\neg Acc(a)$ $\neg Acc(c)$ $Acc(b) \rightarrow ((Val(\beta) \wedge Acc(c)) \vee (Val(\delta) \wedge Acc(d)))$ $Val(\alpha) \rightarrow (Val(\gamma) \wedge Acc(e))$	$\}$
$\Sigma_r(G)$	$= \Sigma(G) \cup \{$	$Acc(d)$ $Acc(e)$ $((Val(\beta) \wedge Acc(c)) \vee (Val(\delta) \wedge Acc(d))) \rightarrow Acc(b)$ $Val(\beta)$ $Val(\delta)$ $Val(\gamma)$ $(Val(\gamma) \wedge Acc(e)) \rightarrow Val(\alpha)$	$\}$
$\Sigma_s(G)$	$= \Sigma(G) \cup \{$	$Acc(d)$ $Acc(e)$ $\neg Acc(a) \rightarrow (Val(\delta) \wedge Acc(d))$ $\neg Acc(b) \rightarrow (Val(\alpha) \wedge Acc(a))$ $\neg Acc(c) \rightarrow (Val(\gamma) \wedge Acc(e))$ $Val(\beta)$ $Val(\delta)$ $Val(\gamma)$ $\neg Val(\alpha) \rightarrow (Val(\beta) \wedge Acc(c))$	$\}$

Using definition of Section 6.2 on page 54, some specific structures of this graph are displayed in the following table:

Semantics	*Structures*
Admissible	$(\varnothing, \{\beta, \delta, \gamma\})$, $(\{d\}, \{\beta, \delta, \gamma\})$, $(\{b, d\}, \{\beta, \delta, \gamma\})$, $(\{e\}, \{\beta, \delta, \gamma\})$, $(\{e\}, \{\alpha, \beta, \delta, \gamma\})$, $(\{d, e\}, \{\beta, \delta, \gamma\})$, $(\{d, e\}, \{\alpha, \beta, \delta, \gamma\})$, $(\{b, d, e\}, \{\beta, \delta, \gamma\})$, $(\{b, d, e\}, \{\alpha, \beta, \delta, \gamma\})$
Stable	$(\{b, d, e\}, \{\alpha, \beta, \delta, \gamma\})$
Complete	$(\{b, d, e\}, \{\alpha, \beta, \delta, \gamma\})$

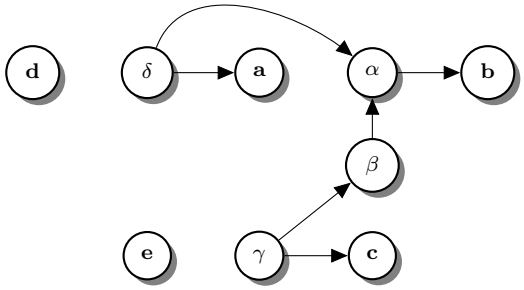
In the case when recursive attacks are encoded with meta-arguments, as in [15], the graph is turned into (note that as δ and γ are not attacked,they can be handled as a simple attack):



The standard Dung extensions of this new graph are displayed in the following table:

Semantics	*Extensions*
Grounded	$\{b, d, e, N_{a\alpha}, N_{c\beta}, \delta, \gamma\}$
Preferred	$\{b, d, e, N_{a\alpha}, N_{c\beta}, \delta, \gamma\}$
Stable	$\{b, d, e, N_{a\alpha}, N_{c\beta}, \delta, \gamma\}$
Complete	$\{b, d, e, N_{a\alpha}, N_{c\beta}, \delta, \gamma\}$

With [6, 21], the graph is turned into:

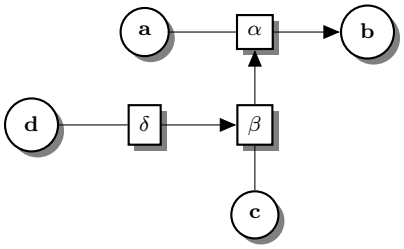


The standard Dung extensions of this new graph are displayed in the following table:

Semantics	*Extensions*
Grounded	$\{b, d, e, \delta, \gamma\}$
Preferred	$\{b, d, e, \delta, \gamma\}$
Stable	$\{b, d, e, \delta, \gamma\}$
Complete	$\{b, d, e, \delta, \gamma\}$

A.2.6 Example 19

Ex. 19

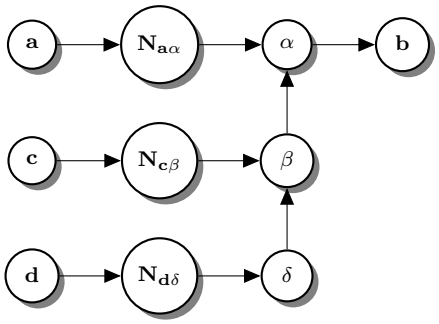


$\Sigma(G)$	$= \{$	$Gr(\alpha) \leftrightarrow Acc(a)$ $Gr(\beta) \leftrightarrow Acc(c)$ $Gr(\delta) \leftrightarrow Acc(d)$ $(Val(\beta) \wedge Acc(c)) \rightarrow \neg Val(\alpha)$ $(Val(\delta) \wedge Acc(d)) \rightarrow \neg Val(\beta)$ $Act(\alpha) \leftrightarrow (Gr(\alpha) \wedge Val(\alpha))$ $Act(\beta) \leftrightarrow (Gr(\beta) \wedge Val(\beta))$ $Act(\delta) \leftrightarrow (Gr(\delta) \wedge Val(\delta))$ $(Val(\alpha) \wedge Acc(a)) \rightarrow NAcc(b)$ $NAcc(b) \rightarrow \neg Acc(b)$	$\}$
$\Sigma_d(G)$	$= \Sigma(G) \cup \{$	$Acc(b) \rightarrow (Val(\beta) \wedge Acc(c))$ $\neg Val(\beta)$ $Val(\alpha) \rightarrow (Val(\delta) \wedge Acc(d))$	$\}$
$\Sigma_r(G)$	$= \Sigma(G) \cup \{$	$Acc(a)$ $Acc(c)$ $Acc(d)$ $(Val(\beta) \wedge Acc(c)) \rightarrow Acc(b)$ $Val(\delta)$ $(Val(\delta) \wedge Acc(d)) \rightarrow Val(\alpha)$	$\}$
$\Sigma_s(G)$	$= \Sigma(G) \cup \{$	$Acc(a)$ $Acc(c)$ $Acc(d)$ $\neg Acc(b) \rightarrow (Val(\alpha) \wedge Acc(a))$ $Val(\delta)$ $\neg Val(\alpha) \rightarrow (Val(\beta) \wedge Acc(c))$ $\neg Val(\beta) \rightarrow (Val(\delta) \wedge Acc(d))$	$\}$

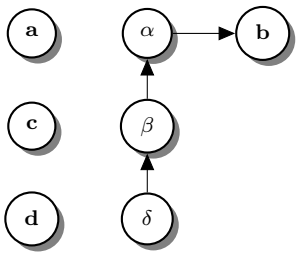
Using definition of Section 6.2 on page 54, some specific structures of this graph are displayed in the following table:

Semantics	*Structures*
Admissible	$(\varnothing, \{\delta\})$, $(\{a\}, \{\delta\})$, $(\{c\}, \{\delta\})$, $(\{d\}, \{\delta\})$ $(\{d\}, \{\alpha, \delta\})$, $(\{a, c\}, \{\delta\})$, $(\{a, d\}, \{\delta\})$, $(\{a, d\}, \{\alpha, \delta\})$, $(\{c, d\}, \{\delta\})$, $(\{c, d\}, \{\alpha, \delta\})$, $(\{a, c, d\}, \{\delta\})$, $(\{a, c, d\}, \{\alpha, \delta\})$
Stable	$(\{a, c, d\}, \{\alpha, \delta\})$
Complete	$(\{a, c, d\}, \{\alpha, \delta\})$

In the case when recursive attacks are encoded with meta-arguments, as in [15], the graph is turned into:



With [6, 21], the graph is turned into:

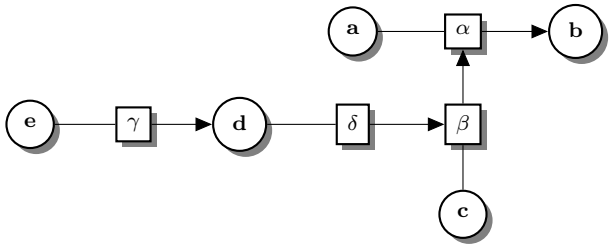


For both cases, the standard Dung extensions of these new graph are displayed in the following table:

Semantics	*Extensions*
Grounded	$\{a, c, d, \delta, \alpha\}$
Preferred	$\{a, c, d, \delta, \alpha\}$
Stable	$\{a, c, d, \delta, \alpha\}$
Complete	$\{a, c, d, \delta, \alpha\}$

A.2.7 Example 20

Ex. 20

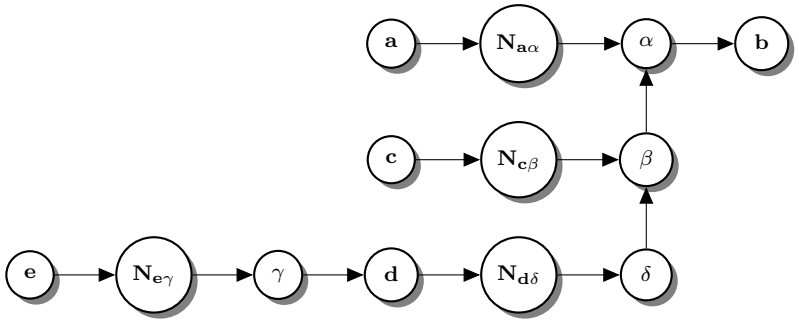


$\Sigma(G)$	$= \{$	$Gr(\alpha) \leftrightarrow Acc(a)$ $Gr(\beta) \leftrightarrow Acc(c)$ $Gr(\delta) \leftrightarrow Acc(d)$ $Gr(\gamma) \leftrightarrow Acc(e)$ $(Val(\beta) \wedge Acc(c)) \rightarrow \neg Val(\alpha)$ $(Val(\delta) \wedge Acc(d)) \rightarrow \neg Val(\beta)$ $Act(\alpha) \leftrightarrow (Gr(\alpha) \wedge Val(\alpha))$ $Act(\beta) \leftrightarrow (Gr(\beta) \wedge Val(\beta))$ $Act(\delta) \leftrightarrow (Gr(\delta) \wedge Val(\delta))$ $Act(\gamma) \leftrightarrow (Gr(\gamma) \wedge Val(\gamma))$ $(Val(\alpha) \wedge Acc(a)) \rightarrow NAcc(b)$ $NAcc(b) \rightarrow \neg Acc(b)$ $(Val(\gamma) \wedge Acc(e)) \rightarrow NAcc(d)$ $NAcc(d) \rightarrow \neg Acc(d)$	$\}$
$\Sigma_d(G)$	$= \Sigma(G) \cup \{$	$Acc(b) \rightarrow (Val(\beta) \wedge Acc(c))$ $\neg Acc(d)$ $Val(\alpha) \rightarrow (Val(\delta) \wedge Acc(d))$ $Val(\beta) \rightarrow (Val(\gamma) \wedge Acc(e))$	$\}$
$\Sigma_r(G)$	$= \Sigma(G) \cup \{$	$Acc(a)$ $Acc(c)$ $Acc(e)$ $(Val(\beta) \wedge Acc(c)) \rightarrow Acc(b)$ $Val(\delta)$ $Val(\gamma)$ $(Val(\delta) \wedge Acc(d)) \rightarrow Val(\alpha)$ $(Val(\gamma) \wedge Acc(e)) \rightarrow Val(\beta)$	$\}$
$\Sigma_s(G)$	$= \Sigma(G) \cup \{$	$Acc(a)$ $Acc(c)$ $Acc(e)$ $\neg Acc(b) \rightarrow (Val(\alpha) \wedge Acc(a))$ $\neg Acc(d) \rightarrow (Val(\gamma) \wedge Acc(e))$ $Val(\delta)$ $Val(\gamma)$ $\neg Val(\alpha) \rightarrow (Val(\beta) \wedge Acc(c))$ $\neg Val(\beta) \rightarrow (Val(\delta) \wedge Acc(d))$	$\}$

Using definition of Section 6.2 on page 54, some specific structures of this graph are displayed in the following table:

Semantics	*Structures*
Admissible	$(\varnothing, \{\delta, \gamma\})$, $(\{a\}, \{\delta, \gamma\})$, $(\{c\}, \{\delta, \gamma\})$, $(\{a, c\}, \{\delta, \gamma\})$ $(\{e\}, \{\delta, \gamma\})$, $(\{e\}, \{\beta, \delta, \gamma\})$, $(\{a, e\}, \{\delta, \gamma\})$, $(\{a, e\}, \{\beta, \delta, \gamma\})$, $(\{c, e\}, \{\delta, \gamma\})$, $(\{c, e\}, \{\beta, \delta, \gamma\})$, $(\{a, c, e\}, \{\delta, \gamma\})$, $(\{a, c, e\}, \{\beta, \delta, \gamma\})$, $(\{b, c, e\}, \{\beta, \delta, \gamma\})$, $(\{a, b, c, e\}, \{\beta, \delta, \gamma\})$
Stable	$(\{a, b, c, e\}, \{\beta, \delta, \gamma\})$
Complete	$(\{a, b, c, e\}, \{\beta, \delta, \gamma\})$

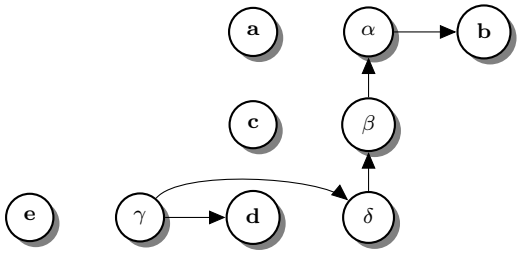
In the case when recursive attacks are encoded with meta-arguments, as in [15], the graph is turned into (note that as γ is not attacked, it can be handled as a simple attack):



The standard Dung extensions of this new graph are displayed in the following table:

Semantics	*Extensions*
Grounded	$\{a, c, e, \gamma, N_{d\delta}, \beta, b\}$
Preferred	$\{a, c, e, \gamma, N_{d\delta}, \beta, b\}$
Stable	$\{a, c, e, \gamma, N_{d\delta}, \beta, b\}$
Complete	$\{a, c, e, \gamma, N_{d\delta}, \beta, b\}$

With [6, 21], the graph is turned into:

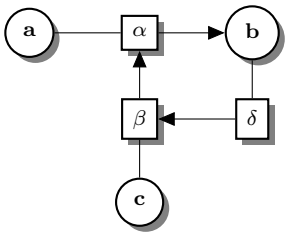


The standard Dung extensions of this new graph are displayed in the following table (similar to those obtained with MAS approach):

Semantics	*Extensions*
Grounded	$\{a, c, e, \gamma, \beta, b\}$
Preferred	$\{a, c, e, \gamma, \beta, b\}$
Stable	$\{a, c, e, \gamma, \beta, b\}$
Complete	$\{a, c, e, \gamma, \beta, b\}$

A.2.8 Example 21

Ex. 21

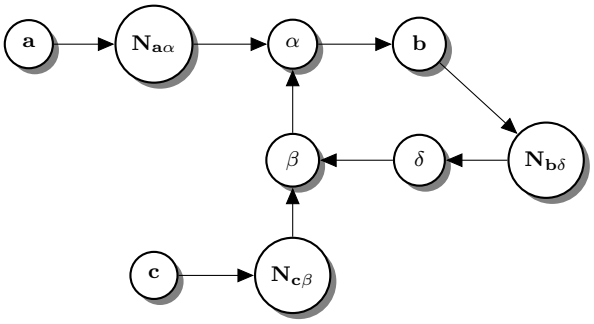


$\Sigma(G)$	$= \{$	$Gr(\alpha) \leftrightarrow Acc(a)$ $Gr(\beta) \leftrightarrow Acc(c)$ $Gr(\delta) \leftrightarrow Acc(b)$ $(Val(\beta) \wedge Acc(c)) \rightarrow \neg Val(\alpha)$ $(Val(\delta) \wedge Acc(b)) \rightarrow \neg Val(\beta)$ $Act(\alpha) \leftrightarrow (Gr(\alpha) \wedge Val(\alpha))$ $Act(\beta) \leftrightarrow (Gr(\beta) \wedge Val(\beta))$ $Act(\delta) \leftrightarrow (Gr(\delta) \wedge Val(\delta))$ $(Val(\alpha) \wedge Acc(a)) \rightarrow NAcc(b)$ $NAcc(b) \rightarrow \neg Acc(b)$	$\}$
$\Sigma_d(G)$	$= \Sigma(G) \cup \{$	$Acc(b) \rightarrow (Val(\beta) \wedge Acc(c))$ $Val(\alpha) \rightarrow (Val(\delta) \wedge Acc(b))$ $Val(\beta) \rightarrow (Val(\alpha) \wedge Acc(a))$	$\}$
$\Sigma_r(G)$	$= \Sigma(G) \cup \{$	$Acc(a)$ $Acc(c)$ $(Val(\beta) \wedge Acc(c)) \rightarrow Acc(b)$ $Val(\delta)$ $(Val(\delta) \wedge Acc(b)) \rightarrow Val(\alpha)$ $(Val(\alpha) \wedge Acc(a)) \rightarrow Val(\beta)$	$\}$
$\Sigma_s(G)$	$= \Sigma(G) \cup \{$	$Acc(a)$ $Acc(c)$ $\neg Acc(b) \rightarrow (Val(\alpha) \wedge Acc(a))$ $Val(\delta)$ $\neg Val(\alpha) \rightarrow (Val(\beta) \wedge Acc(c))$ $\neg Val(\beta) \rightarrow (Val(\delta) \wedge Acc(b))$	$\}$

Using definition of Section 6.2 on page 54, some specific structures of this graph are displayed in the following table:

Semantics	*Structures*
Admissible	$(\varnothing, \{\delta\})$, $(\{a\}, \{\delta\})$, $(\{c\}, \{\delta\})$, $(\{a, c\}, \{\delta\})$
Stable	$\nexists$
Complete	$(\{a, c\}, \{\delta\})$

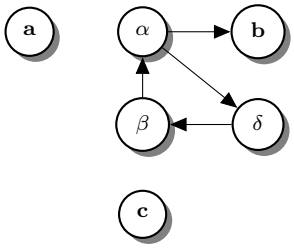
In the case when recursive attacks are encoded with meta-arguments, as in [15], the graph is turned into:



The standard Dung extensions of this new graph are displayed in the following table:

Semantics	*Extensions*
Grounded	$\{a, c\}$
Preferred	$\{a, c\}$
Stable	$\nexists$
Complete	$\{a, c\}$

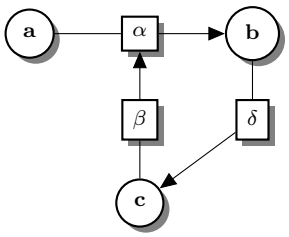
With [6, 21], the graph is turned into:



The standard Dung extensions of this new graph are exactly those obtained with MAS approach.

A.2.9 Example 22

Ex. 22

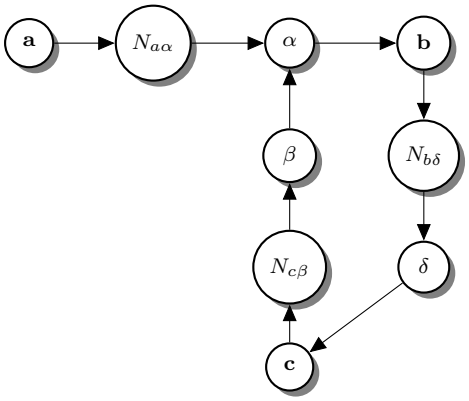


$\Sigma(G)$	$= \{$	$Gr(\alpha) \leftrightarrow Acc(a)$ $Gr(\beta) \leftrightarrow Acc(c)$ $Gr(\delta) \leftrightarrow Acc(b)$ $(Val(\beta) \wedge Acc(c)) \rightarrow \neg Val(\alpha)$ $Act(\alpha) \leftrightarrow (Gr(\alpha) \wedge Val(\alpha))$ $Act(\beta) \leftrightarrow (Gr(\beta) \wedge Val(\beta))$ $Act(\delta) \leftrightarrow (Gr(\delta) \wedge Val(\delta))$ $(Val(\alpha) \wedge Acc(a)) \rightarrow NAcc(b)$ $NAcc(b) \rightarrow \neg Acc(b)$ $(Val(\delta) \wedge Acc(b)) \rightarrow NAcc(c)$ $NAcc(c) \rightarrow \neg Acc(c)$	$\}$
$\Sigma_d(G)$	$= \Sigma(G) \cup \{$	$Acc(b) \rightarrow (Val(\beta) \wedge Acc(c))$ $Acc(c) \rightarrow (Val(\alpha) \wedge Acc(a))$ $Val(\alpha) \rightarrow (Val(\delta) \wedge Acc(b))$	$\}$
$\Sigma_r(G)$	$= \Sigma(G) \cup \{$	$Acc(a)$ $(Val(\beta) \wedge Acc(c)) \rightarrow Acc(b)$ $(Val(\alpha) \wedge Acc(a)) \rightarrow Acc(c)$ $Val(\beta)$ $Val(\delta)$ $(Val(\delta) \wedge Acc(b)) \rightarrow Val(\alpha)$	$\}$
$\Sigma_s(G)$	$= \Sigma(G) \cup \{$	$Acc(a)$ $\neg Acc(b) \rightarrow (Val(\alpha) \wedge Acc(a))$ $\neg Acc(c) \rightarrow (Val(\delta) \wedge Acc(b))$ $Val(\beta)$ $Val(\delta)$ $\neg Val(\alpha) \rightarrow (Val(\beta) \wedge Acc(c))$	$\}$

Using definition of Section 6.2 on page 54, some specific structures of this graph are displayed in the following table:

Semantics	*Structures*
Admissible	$(\varnothing, \{\beta, \delta\})$, $(\{a\}, \{\beta, \delta\})$
Stable	$\nexists$
Complete	$(\{a\}, \{\beta, \delta\})$

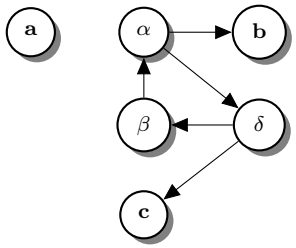
In the case when recursive attacks are encoded with meta-arguments, as in [15], the graph is turned into:



The standard Dung extensions of this new graph are displayed in the following table:

Semantics	*Extensions*
Grounded	$\{a\}$
Preferred	$\{a\}$
Stable	$\nexists$
Complete	$\{a\}$

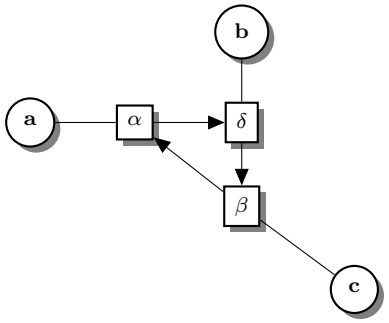
With [6, 21], the graph is turned into:



The standard Dung extensions of this new graph are exactly those obtained with MAS approach.

A.2.10 Example 23

Ex. 23

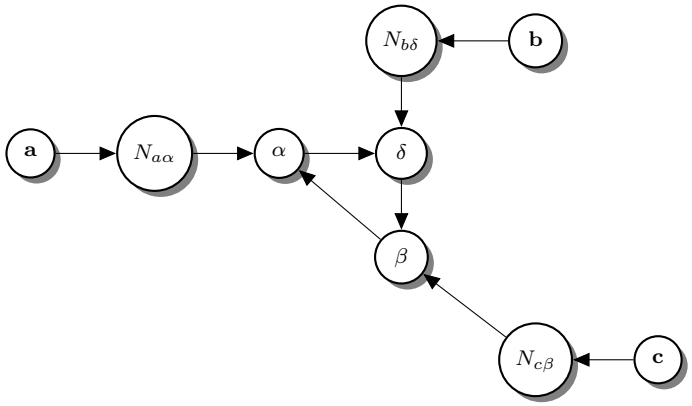


$\Sigma(G)$	$= \{$	$Gr(\alpha) \leftrightarrow Acc(a)$ $Gr(\beta) \leftrightarrow Acc(c)$ $Gr(\delta) \leftrightarrow Acc(b)$ $(Val(\beta) \wedge Acc(c)) \rightarrow \neg Val(\alpha)$ $(Val(\delta) \wedge Acc(b)) \rightarrow \neg Val(\beta)$ $(Val(\alpha) \wedge Acc(a)) \rightarrow \neg Val(\delta)$ $Act(\alpha) \leftrightarrow (Gr(\alpha) \wedge Val(\alpha))$ $Act(\beta) \leftrightarrow (Gr(\beta) \wedge Val(\beta))$ $Act(\delta) \leftrightarrow (Gr(\delta) \wedge Val(\delta))$	$\}$
$\Sigma_d(G)$	$= \Sigma(G) \cup \{$	$Val(\alpha) \rightarrow (Val(\delta) \wedge Acc(b))$ $Val(\beta) \rightarrow (Val(\alpha) \wedge Acc(a))$ $Val(\delta) \rightarrow (Val(\beta) \wedge Acc(c))$	$\}$
$\Sigma_r(G)$	$= \Sigma(G) \cup \{$	$Acc(a)$ $Acc(b)$ $Acc(c)$ $(Val(\delta) \wedge Acc(b)) \rightarrow Val(\alpha)$ $(Val(\alpha) \wedge Acc(a)) \rightarrow Val(\beta)$ $(Val(\beta) \wedge Acc(c)) \rightarrow Val(\delta)$	$\}$
$\Sigma_s(G)$	$= \Sigma(G) \cup \{$	$Acc(a)$ $Acc(b)$ $Acc(c)$ $\neg Val(\alpha) \rightarrow (Val(\beta) \wedge Acc(c))$ $\neg Val(\beta) \rightarrow (Val(\delta) \wedge Acc(b))$ $\neg Val(\delta) \rightarrow (Val(\alpha) \wedge Acc(a))$	$\}$

Using definition of Section 6.2 on page 54, some specific structures of this graph are displayed in the following table:

Semantics	*Structures*
Admissible	$(\varnothing, \varnothing)$, $(\{a\}, \varnothing)$, $(\{b\}, \varnothing)$, $(\{c\}, \varnothing)$, $(\{a, b\}, \varnothing)$, $(\{a, c\}, \varnothing)$, $(\{b, c\}, \varnothing)$, $(\{a, b, c\}, \varnothing)$
Stable	$\nexists$
Complete	$(\{a, b, c\}, \varnothing)$

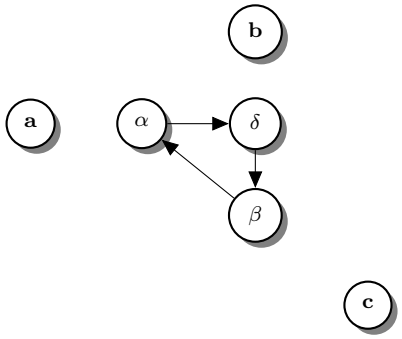
In the case when recursive attacks are encoded with meta-arguments, as in [15], the graph is turned into:



The standard Dung extensions of this new graph are displayed in the following table:

Semantics	*Extensions*
Grounded	$\{a, b, c\}$
Preferred	$\{a, b, c\}$
Stable	$\nexists$
Complete	$\{a, b, c\}$

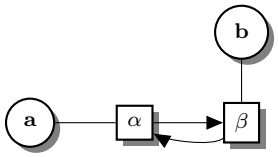
With [6, 21], the graph is turned into:



The standard Dung extensions of this new graph are exaxtly those obtained with MAS approach.

A.2.11 Example 24

Ex. 24

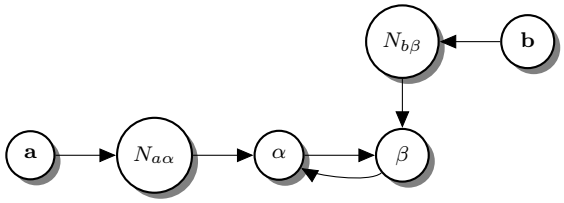


$\Sigma(G)$	$= \{$	$Gr(\alpha) \leftrightarrow Acc(a)$ $Gr(\beta) \leftrightarrow Acc(b)$ $(Val(\beta) \wedge Acc(b)) \rightarrow \neg Val(\alpha)$ $(Val(\alpha) \wedge Acc(a)) \rightarrow \neg Val(\beta)$ $Act(\alpha) \leftrightarrow (Gr(\alpha) \wedge Val(\alpha))$ $Act(\beta) \leftrightarrow (Gr(\beta) \wedge Val(\beta))$	$\}$
$\Sigma_d(G)$	$= \Sigma(G) \cup \{$	$Val(\alpha) \rightarrow Acc(a)$ $Val(\beta) \rightarrow Acc(b)$	$\}$
$\Sigma_r(G)$	$= \Sigma(G) \cup \{$	$Acc(a)$ $Acc(b)$	$\}$
$\Sigma_s(G)$	$= \Sigma(G) \cup \{$	$Acc(a)$ $Acc(b)$ $\neg Val(\alpha) \rightarrow (Val(\beta) \wedge Acc(b))$ $\neg Val(\beta) \rightarrow (Val(\alpha) \wedge Acc(a))$	$\}$

Using definition of Section 6.2 on page 54, some specific structures of this graph are displayed in the following table:

Semantics	*Structures*
Admissible	$(\varnothing, \varnothing)$, $(\{a\}, \varnothing)$, $(\{a\}, \{\alpha\})$, $(\{b\}, \varnothing)$, $(\{b\}, \{\beta\})$, $(\{a, b\}, \varnothing)$, $(\{a, b\}, \{\alpha\})$, $(\{a, b\}, \{\beta\})$
Stable	$(\{a, b\}, \{\alpha\})$, $(\{a, b\}, \{\beta\})$
Complete	$(\{a, b\}, \varnothing)$, $(\{a, b\}, \{\alpha\})$, $(\{a, b\}, \{\beta\})$

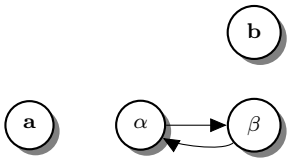
In the case when recursive attacks are encoded with meta-arguments, as in [15], the graph is turned into:



The standard Dung extensions of this new graph are displayed in the following table:

Semantics	*Extensions*
Grounded	$\{a, b\}$
Preferred	$\{a, b, \alpha\}$, $\{a, b, \beta\}$
Stable	$\{a, b, \alpha\}$, $\{a, b, \beta\}$
Complete	$\{a, b\}$, $\{a, b, \alpha\}$, $\{a, b, \beta\}$

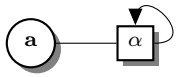
With [6, 21], the graph is turned into:



The standard Dung extensions of this new graph are exactly those obtained with MAS approach.

A.2.12 Example 25

Ex. 25

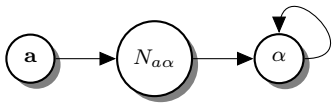


$\Sigma(G)$	$= \{$	$Gr(\alpha) \leftrightarrow Acc(a)$ $(Val(\alpha) \wedge Acc(a)) \rightarrow$ $Act(\alpha) \leftrightarrow (Gr(\alpha) \wedge Val(\alpha))$	$\}$
$\Sigma_d(G)$	$= \Sigma(G) \cup \{$	$Val(\alpha) \rightarrow Acc(a)$	$\}$
$\Sigma_r(G)$	$= \Sigma(G) \cup \{$	$Acc(a)$	$\}$
$\Sigma_s(G)$	$= \Sigma(G) \cup \{$	$Acc(a)$ $Val(\alpha)$	$\}$

Using definition of Section 6.2 on page 54, some specific structures of this graph are displayed in the following table:

Semantics	*Structures*
Admissible	$(\varnothing, \varnothing)$, $(\{a\}, \varnothing)$
Stable	$\nexists$
Complete	$(\{a\}, \varnothing)$

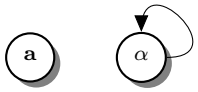
In the case when recursive attacks are encoded with meta-arguments, as in [15], the graph is turned into:



The standard Dung extensions of this new graph are displayed in the following table:

Semantics	*Extensions*
Grounded	$\{a\}$
Preferred	$\{a\}$
Stable	$\nexists$
Complete	$\{a\}$

With [6, 21], the graph is turned into:



The standard Dung extensions of this new graph are exactly those obtained with MAS approach.

Appendix B

Proofs

B.1 Proofs of Section 4.1.2 on page 19

Proof of Prop. 1 on page 20: A model $\mathcal{I}$ of $\Sigma^0(G)$ can be defined as follows:
$\forall x \in \mathbf{A}, \mathcal{I}(Acc(x)) =$ *false* and $\mathcal{I}(NAcc(x)) =$ *true*. □

B.2 Proofs of Section 4.2.2 on page 24

Proof of Prop. 2 on page 27: A model $\mathcal{I}$ of $\Sigma(G)$ can be defined as follows:
$\forall x \in \mathbf{A}, \mathcal{I}(Acc(x)) =$ *false* and $\mathcal{I}(NAcc(x)) =$ *true*.
$\forall \alpha \in \mathbf{R}, \mathcal{I}(Gr(\alpha)) = \mathcal{I}(Act(\alpha)) =$ *false* and $\mathcal{I}(Val(\alpha)) =$ *true*. □

B.3 Proofs of Section 5.1 on page 33

Proof of Prop. 3 on page 35: The model $\mathcal{I}$ of $\Sigma^0(G)$ given in the proof of Proposition 1 on page 20 can also be used for proving the consistency of $\Sigma^0_d(G)$:
$\forall x \in \mathbf{A}, \mathcal{I}(Acc(x)) =$ *false* and $\mathcal{I}(NAcc(x)) =$ *true*. □

Proof of Prop. 4 on page 36: A model $\mathcal{I}$ of $\Sigma^0_r(G)$ can be defined using the following iterative construction:

1. Let $M_0 = \{Acc(x) |\ x$ is unattacked $\}$.
2. $\forall j \geq 0$, let $M_{j+1} = \{Acc(x) | \exists \Phi \in \Sigma^0_r(G), \Phi = (\wedge_{i=1..k}(\vee_{a \in B_i} Acc(a))) \rightarrow Acc(x)$ and for each b_i (attacker of x), $\exists a \in B_i$ such that $Acc(a) \in (M_0 \cup \ldots M_j)\}$.
3. Let $M = \bigcup M_j, j \geq 0$. Then $\mathcal{I}$ is defined by $\mathcal{I}(Acc(x)) =$ *true* if and only if $Acc(x) \in M$ and $\mathcal{I}(NAcc(x)) =$ *false* if and only if $\mathcal{I}(Acc(x)) =$ *true*.

Obviously, $\mathcal{I}$ satisfies the formulae issued from $\mathbf{P}_{\mathbf{reins}}$.
Moreover, the definition of M exactly corresponds to the iterative construction of the least fixed

point of the characteristic function $\mathcal{F}$ of G. So M is exactly the set $Acc(GE)$, where GE denotes the grounded extension of G. We know that GE is conflict-free (see section 2.2 on page 7), so following Proposition 6 on page 39, M is consistent with $\Sigma^0(G)$. Thus $\mathcal{I}$ is a model of $\Sigma^0_r(G)$.

□

Proof of Prop. 5 on page 39:

⇒ Assume that S is conflict-free. Let us define an interpretation $\mathcal{I}$ of $\Sigma^0(G)$ as follows : $\mathcal{I}(Acc(x)) =$ *true* if and only if $x \in S$ and $\mathcal{I}(NAcc(x)) =$ *true* if and only if $\mathcal{I}(Acc(x)) =$ *false*. We have $S_\mathcal{I} = S$. It remains to prove that $\mathcal{I}$ is a model of $\Sigma^0(G)$. Obviously $\mathcal{I}$ satisfies the formula $NAcc(x) \rightarrow \neg Acc(x)$. So, if $\mathcal{I}$ is not a model of $\Sigma^0(G)$, there exists a formula $Acc(a) \rightarrow NAcc(b)$ that is not satisfied by $\mathcal{I}$. In that case $\mathcal{I}(Acc(a)) =$ *true* and $\mathcal{I}(NAcc(b)) =$ *false*. By definition of $\mathcal{I}$, we also have $\mathcal{I}(Acc(b)) =$ *true*, and a, b are in S. The formula $Acc(a) \rightarrow NAcc(b)$ encodes an attack from a to b. So that is in contradiction with S being conflict-free.

⇐ Let $\mathcal{I}$ be a model of $\Sigma^0(G)$. We prove that $S_\mathcal{I}$ is conflict-free. If it is not the case, there exist two arguments a, b in $S_\mathcal{I}$ such that a attacks b. So $\Sigma^0(G)$ contains the formulas $Acc(a) \rightarrow NAcc(b)$ and $NAcc(b) \rightarrow \neg Acc(b)$. As a, b belong to $S_\mathcal{I}$, $\mathcal{I}(Acc(a)) =$ *true* and $\mathcal{I}(Acc(b)) =$ *true*. That is in contradiction with $\mathcal{I}$ being a model of $\Sigma^0(G)$.

□

Proof of Prop. 6 on page 39:

⇒ Assume that S is conflict-free. From Proposition 5 on page 39, there exists a model $\mathcal{I}$ of $\Sigma^0(G)$ such that $S_\mathcal{I} = S$. Obviously $\mathcal{I}$ is a model of $Acc(S) \cup \Sigma^0(G)$.

⇐ Let $\mathcal{I}$ be a model of $Acc(S) \cup \Sigma^0(G)$. As $\mathcal{I}$ is a model of $Acc(S)$, we have $S \subseteq S_\mathcal{I}$. As $\mathcal{I}$ is a model of $\Sigma^0(G)$, from Proposition 5 on page 39 again, we know that $S_\mathcal{I}$ is conflict-free. So S is conflict-free.

□

Proof of Prop. 7 on page 39: It is a direct consequence of Prop. 5 on page 39.

⇒ Assume that S is a naive extension. From Proposition 5 on page 39, there exists a model $\mathcal{I}$ of $\Sigma^0(G)$ such that $S_\mathcal{I} = S$. If $\mathcal{I}$ is not $Acc(\mathbf{A})$-maximal, there exists $\mathcal{I}'$ model of $\Sigma^0(G)$ such that $S_\mathcal{I} \subset S_{\mathcal{I}'}$. Then $S_{\mathcal{I}'}$ is conflict-free and strictly contains S, which is in contradiction with S being a naive extension.

⇐ Let $\mathcal{I}$ be an $Acc(\mathbf{A})$-maximal model of $\Sigma^0(G)$ such that $S_\mathcal{I} = S$. Then S is conflict-free. If S is not a naive extension, there exists S' conflict-free such that $S \subset S'$. So there exists a model $\mathcal{I}'$ of $\Sigma^0(G)$ such that $S_{\mathcal{I}'} = S'$. Then we have $S_\mathcal{I} = S \subset S_{\mathcal{I}'}$ which is in contradiction with $\mathcal{I}$ being $Acc(\mathbf{A})$-maximal.

□

Proof of Prop. 8 on page 39: It is a direct consequence of Prop. 6 on page 39.

$\Rightarrow$ Assume that S is a naive extension. We know that $Acc(S) \cup \Sigma^0(G)$ is consistent. Assume that there exists $L \subseteq Acc(\mathbf{A})$ such that $Acc(S) \subset L$ and $L \cup \Sigma^0(G)$ is consistent. Let S' such that $L = Acc(S')$. We have $S \subset S'$ and S' conflict-free (from Proposition 6 on page 39). That is in contradiction with S being a naive extension. So $Acc(S)$ is a maximal (for set-inclusion) subset of $Acc(\mathbf{A})$ consistent with $\Sigma^0(G)$.

$\Leftarrow$ Assume that $Acc(S)$ is a maximal (for set-inclusion) subset of $Acc(\mathbf{A})$ consistent with $\Sigma^0(G)$. From Proposition 6 on page 39, S is conflict-free. If there exists S' such that $S \subset S'$ and S' conflict-free, $Acc(S') \cup \Sigma^0(G)$ is consistent and $Acc(S) \subset Acc(S')$, which is in contradiction with the assumption on $Acc(S)$.

□

Proof of Prop. 9 on page 39:

$\Rightarrow$ Let S be admissible. S is conflict-free, so due to the proof of Proposition 5 on page 39, $\exists \mathcal{I}$ model of $\Sigma^0(G)$ with $S_{\mathcal{I}} = S$ and such that:

- $\forall x \in S, \mathcal{I}(Acc(x)) =$ *true* and $\mathcal{I}(NAcc(x)) =$ *false*;
- $\forall x \notin S, \mathcal{I}(Acc(x)) =$ *false* and $\mathcal{I}(NAcc(x)) =$ *true*.

We prove that $\mathcal{I}$ is a model of $\Sigma^0_d(G)$. We just have to consider the formulae issued from Property $\mathbf{P_{def}}$. Two kinds of formulae must be considered:

- A formula $\Phi = \neg Acc(x)$ corresponds to the particular case when x is attacked by an argument b which is unattacked. In that case, x cannot belong to any admissible set, so $x \notin S$ and $\mathcal{I}(Acc(x)) =$ *false*. So $\mathcal{I}$ satisfies Φ.
- A formula $\Phi = (Acc(x) \rightarrow (Acc(a_1) \vee \ldots \vee Acc(a_k))$ corresponds to the case when x is attacked by b, $\{a_1, \ldots a_k\}$ being the set of attackers of b.
  - either $x \in S$; since S is admissible, there exists i such that $a_i \in S$. Then $\mathcal{I}(Acc(x)) = \mathcal{I}(Acc(a_i)) =$ *true*;
  - or $x \notin S$ then $\mathcal{I}(Acc(x)) =$ *false*.

  In both cases, Φ is satisfied by $\mathcal{I}$.

In conclusion, each formula of $\Sigma^0_d(G)$ is satisfied by $\mathcal{I}$. So $\mathcal{I}$ is a model of $\Sigma^0_d(G)$.

$\Leftarrow$ We prove that if $\mathcal{I}$ is a model of $\Sigma^0_d(G)$, then $S_{\mathcal{I}}$ is admissible.

1. As $\Sigma^0_d(G)$ contains $\Sigma^0(G)$, if $\mathcal{I}$ is a model of $\Sigma^0_d(G)$, $\mathcal{I}$ is also a model of $\Sigma^0(G)$ and then $S_{\mathcal{I}}$ is conflict-free (due to Proposition 5 on page 39).
2. Let us prove that $S_{\mathcal{I}} \subseteq \mathcal{F}(S_{\mathcal{I}})$.
   By definition, $\mathcal{F}(S_{\mathcal{I}}) = \{x \in \mathbf{A} | x$ is acceptable wrt $S_{\mathcal{I}}\}$ = $\{x \in \mathbf{A} | \forall b \in \mathbf{A}$ such that $b\mathbf{R}x, \exists a \in S_{\mathcal{I}}$ and $a\mathbf{R}b\}$.
   If $x \in S_{\mathcal{I}}$ is unattacked, obviously $x \in \mathcal{F}(S_{\mathcal{I}})$.
   So let us consider $x \in S_{\mathcal{I}}$ such that x is attacked. Let b be any attacker of x. $\Sigma^0_d(G)$ contains the formulae $Acc(b) \rightarrow NAcc(x)$ and $NAcc(x) \rightarrow \neg Acc(x)$. Since $\mathcal{I}(Acc(x)) =$ *true*, and $\mathcal{I}$ is a model of $\Sigma^0_d(G)$, then $\mathcal{I}(NAcc(x)) = \mathcal{I}(Acc(b)) =$ *false*. Two different cases must be considered for b:
   - If b is unattacked, $\Sigma^0_d(G)$ contains a formula equivalent to $\neg Acc(x)$, which is in contradiction with $\mathcal{I}$ being a model of $\Sigma^0_d(G)$ and $\mathcal{I}(Acc(x)) =$ *true*. So this case is impossible.

- If b is attacked, let $a_1, \ldots a_k$ be the attackers of b.
  $\Sigma_d^0(G)$ contains the formula $Acc(x) \rightarrow (Acc(a_1) \vee \ldots \vee Acc(a_k))$. Since $\mathcal{I}(Acc(x)) =$ ***true***, and $\mathcal{I}$ is a model of $\Sigma_d^0(G)$, then $\mathcal{I}(Acc(a_i)) =$ ***true*** for at least one a_i. It means that $a_i \in S_{\mathcal{I}}$. So $x \in \mathcal{F}(S_{\mathcal{I}})$.

□

Proof of Prop. 10 on page 39:

1. Assume S is admissible. From Proposition 9 on page 39, $\exists \mathcal{I}$ model of $\Sigma_d^0(G)$ such that $S = S_{\mathcal{I}}$. Obviously $\mathcal{I}$ is a model of $Acc(S) \cup \Sigma_d^0(G)$.
2. Let $\mathcal{I}$ be a model of $Acc(S) \cup \Sigma_d^0(G)$. As $\mathcal{I}$ is a model of $Acc(S)$, we have $S \subseteq S_{\mathcal{I}}$. As $\mathcal{I}$ is a model of $\Sigma_d^0(G)$, from Proposition 9 on page 39, we know that $S_{\mathcal{I}}$ is admissible. So there is an admissible set containing S.

□

Proof of Prop. 11 on page 40: It is a direct consequence of Prop. 9 on page 39.

$\Rightarrow$ Let S be a preferred extension. From Proposition 9 on page 39, there exists a model $\mathcal{I}$ of $\Sigma_d^0(G)$ such that $S = S_{\mathcal{I}}$. If $\mathcal{I}$ is not $Acc(\mathbf{A})$-maximal, there exists $\mathcal{I}'$ model of $\Sigma_d^0(G)$ such that $S_{\mathcal{I}} \subset S_{\mathcal{I}'}$. Following Proposition 9 on page 39 again, $S_{\mathcal{I}'}$ is admissible. So there exists an admissible set $S_{\mathcal{I}'}$ such that $S = S_{\mathcal{I}} \subset S_{\mathcal{I}'}$, which is in contradiction with S being a preferred extension.

$\Leftarrow$ Let $\mathcal{I}$ be an $Acc(\mathbf{A})$-maximal model of $\Sigma_d^0(G)$ such that $S_{\mathcal{I}} = S$. If $S_{\mathcal{I}}$ is not a preferred extension, there exists $S' \subseteq \mathbf{A}$ such that $S_{\mathcal{I}} \subset S'$ and S' is admissible. Following Proposition 9 on page 39, there exists $\mathcal{I}'$ model of $\Sigma_d^0(G)$ such that $S' = S_{\mathcal{I}'}$. So, there is a contradiction with $\mathcal{I}$ being an $Acc(\mathbf{A})$-maximal model of $\Sigma_d^0(G)$.

□

Proof of Prop. 12 on page 40:

$\Rightarrow$ Assume that S is a preferred extension. S is admissible, so from Proposition 10 on page 39, we know that $Acc(S) \cup \Sigma_d^0(G)$ is consistent. Assume that there exists $L \subseteq Acc(\mathbf{A})$ such that $Acc(S) \subset L$ and $L \cup \Sigma_d^0(G)$ is consistent. Let S' such that $L = Acc(S')$. We have $S \subset S'$ and $Acc(S') \cup \Sigma_d^0(G)$ is consistent. From Proposition 10 on page 39 again, we know that $S' \subseteq S''$ where S'' is admissible. So $S \subset S''$, which is in contradiction with S being a preferred extension.

$\Leftarrow$ Assume that $Acc(S)$ is a maximal (for set-inclusion) subset of $Acc(\mathbf{A})$ consistent with $\Sigma_d^0(G)$. We have to prove that S is admissible and then that S is maximal admissible.
From Proposition 10 on page 39, we know that $S \subseteq S'$ where S' is admissible. Then we have $Acc(S') \cup \Sigma_d^0(G)$ is consistent. As $Acc(S)$ is a maximal (for set-inclusion) subset of $Acc(\mathbf{A})$ consistent with $\Sigma_d^0(G)$, we have $Acc(S) = Acc(S')$ and $S = S'$. So S is admissible.
If S is not a preferred extension, there exists an admissible set S'' strictly containing S. Then we know that $Acc(S'') \cup \Sigma_d^0(G)$ is consistent, which is in contradiction with $Acc(S)$ being a maximal (for set-inclusion) subset of $Acc(\mathbf{A})$ consistent with $\Sigma_d^0(G)$. So we have proved that S is a preferred extension.

□

Proof of Prop. 13 on page 40: It follows from the proof of Proposition 4 on page 36.
Let M be the subset of $Acc(\mathbf{A})$ built in the proof of Proposition 4 on page 36, with an iterative procedure. We know that $M = Acc(GE)$, or equivalently $GE = \{x \in \mathbf{A} | Acc(x) \in M\}$, where GE denotes the grounded extension of G.
Let $\mathcal{I}$ be the model of $\Sigma_r^0(G)$ associated with M in the proof of Proposition 4 on page 36. We have $x \in S_\mathcal{I}$ if and only if $Acc(x) \in M$, so $S_\mathcal{I} = \{x \in \mathbf{A} | Acc(x) \in M\} = GE$.
From the definition of M, it is easy to see that each model $\mathcal{J}$ of the formulae issued from $\mathbf{P}_{\mathbf{reins}}$ must satisfy the literals of M, or equivalently $\{x \in \mathbf{A} | Acc(x) \in M\} \subseteq S_\mathcal{J}$.
As a consequence, it is easy to see that for each $\mathcal{J}$ an $Acc(\mathbf{A})$-minimal model of $\Sigma_r^0(G)$, we have $S_\mathcal{J} = \{x \in \mathbf{A} | Acc(x) \in M\} = GE$. Note that $\mathcal{I}$ itself is an $Acc(\mathbf{A})$-minimal model of $\Sigma_r^0(G)$.
□

Proof of Prop. 14 on page 40: Let $\mathcal{I}$ be an $Acc(\mathbf{A})$-minimal model of $\Sigma_r^0(G)$. We know that $S_\mathcal{I} = GE$.

- We first prove that $GE = S_\mathcal{I} \subseteq S_g$. Let $x \in S_\mathcal{I}$. By definition of $S_\mathcal{I}$, there is no model $\mathcal{I}'$ of $\Sigma_r^0(G)$ such that $S_{\mathcal{I}'} \subset S_\mathcal{I}$. So the set of formulae $\Sigma_r^0(G) \cup \{\neg Acc(x)\}$ has no model. It means that $\Sigma_r^0(G) \vdash Acc(x)$, and so $x \in S_g$.
- Conversely, let $x \in S_g$. As $\Sigma_r^0(G) \vdash Acc(x)$, $Acc(x)$ is *true* in each model of $\Sigma_r^0(G)$. So $Acc(x)$ is *true* in the particular model $\mathcal{I}$. It means that $x \in S_\mathcal{I} = GE$.

□

Proof of Prop. 15 on page 40:

1. Let $\mathcal{I}$ be a model of $\Sigma_r^0(G)$. $S_\mathcal{I} = \{x \in \mathbf{A} | \mathcal{I}(Acc(x)) = true\}$. We have to prove that $\mathcal{F}(S_\mathcal{I}) \subseteq S_\mathcal{I}$. Let $x \in \mathcal{F}(S_\mathcal{I})$.
   If x is not attacked, the formula $Acc(x)$ belongs to $\Sigma_r^0(G)$. As $\mathcal{I}$ is a model of $\Sigma_r^0(G)$, $\mathcal{I}$ satisfies $Acc(x)$ which means that $x \in S_\mathcal{I}$.
   If x is attacked, let $b_1, \ldots, b_k$ be the attackers of x, and for each i, let B_i be the set of attackers of b_i. $\Sigma_r^0(G)$ contains the formula $\Phi = (\wedge_{i=1..k}(\vee_{a \in B_i} Acc(a))) \rightarrow Acc(x)$. As $x \in \mathcal{F}(S_\mathcal{I})$, for each i, there exists $a \in B_i$ such that $a \in S_\mathcal{I}$. Or equivalently, for each i, there exists $a \in B_i$ such that $\mathcal{I}(Acc(a)) = true$. As $\mathcal{I}$ satisfies the formula Φ, we have $\mathcal{I}(Acc(x)) = true$ which means that $x \in S_\mathcal{I}$.
2. Assume that $\mathcal{F}(S) \subseteq S$ and S is conflict-free. Let us consider the interpretation $\mathcal{I}$ defined by:
   - $\forall x \in S, \mathcal{I}(Acc(x)) =$ *true* and $\mathcal{I}(NAcc(x)) =$ *false*;
   - $\forall x \notin S, \mathcal{I}(Acc(x)) =$ *false* and $\mathcal{I}(NAcc(x)) =$ *true*.

   We have $S_\mathcal{I} = S$ and from the proof of Proposition 5 on page 39, we know that $\mathcal{I}$ is a model of $\Sigma^0(G)$. It remains to prove that $\mathcal{I}$ satisfies the formulae of $\Sigma_r^0(G) \setminus \Sigma^0(G)$. Two kinds of formulae must be considered:
   A formula $\Phi = Acc(x)$ corresponds to the particular case when x is not attacked. In that case, $x \in \mathcal{F}(S) \subseteq S$, so $x \in S$ and then $\mathcal{I}(Acc(x)) =$ *true*. So $\mathcal{I}$ satisfies Φ.
   A formula $\Phi = (\wedge_{i=1..k}(\vee_{a \in B_i} Acc(a))) \rightarrow Acc(x)$ corresponds to the case when $b_1, \ldots, b_k$ are the attackers of x, and for each i, B_i denotes the set of attackers of b_i.

- either $x \in \mathcal{F}(S)$, so for each i, there exists $a \in B_i$ such that $a \in S$. Or equivalently, or each i, there exists $a \in B_i$ such that $\mathcal{I}(Acc(a)) = \textit{true}$. As $\mathcal{F}(S) \subseteq S$, $x \in S$ and then $\mathcal{I}(Acc(x)) = \textit{true}$. In that case the formula Φ is satisfied by $\mathcal{I}$.
- or $x \notin \mathcal{F}(S)$, so there exists i such that for each $a \in B_i, a \notin S$. In other words, there exists i such that for each $a \in B_i, \mathcal{I}(Acc(a)) = \textit{false}$. In that case the formula Φ is trivially satisfied.

In both cases, Φ is satisfied by $\mathcal{I}$.

□

Proof of Prop. 16 on page 40:

⇒ Let S be a complete extension. S is admissible and $\mathcal{F}(S) \subseteq S$. From the proofs of Propositions 9 on page 39 and 15 on page 40, we know that there exists a same model $\mathcal{I}$ of $\Sigma_d^0(G)$ and of $\Sigma_r^0(G)$ such that $S = S_{\mathcal{I}}$. This model is defined by :

- $\forall x \in S, \mathcal{I}(Acc(x)) = \textit{true}$ and $\mathcal{I}(NAcc(x)) = \textit{false}$;
- $\forall x \notin S, \mathcal{I}(Acc(x)) = \textit{false}$ and $\mathcal{I}(NAcc(x)) = \textit{true}$.

⇐ The converse is an immediate consequence of Propositions 9 on page 39 and 15 on page 40.

□

Proof of Prop. 17 on page 41:

⇒ Let S be a complete extension. From Proposition 16 on page 40, there exists $\mathcal{I}$ model of $\Sigma_d^0(G) \cup \Sigma_r^0(G)$ such that $S = S_{\mathcal{I}}$.
By definition of $S_{\mathcal{I}}$, $S = S_{\mathcal{I}}$ means that $\forall x \in S, \mathcal{I}(Acc(x)) = \textit{true}$ and $\forall x \in \mathbf{A} \setminus S, \mathcal{I}(Acc(x)) = \textit{false}$, or equivalently $\forall x \in \mathbf{A} \setminus S, \mathcal{I}(\neg Acc(x)) = \textit{true}$. So $\mathcal{I}$ is a model of $Acc(S) \cup \{\neg Acc(x) | x \in \mathbf{A} \setminus S\} \cup \Sigma_d^0(G) \cup \Sigma_r^0(G)$.
Moreover, from the proofs of the previous propositions, $\mathcal{I}$ can be chosen such that:
$\mathcal{I}(NAcc(x)) = \textit{true}$ if and only if $\mathcal{I}(Acc(x)) = \textit{false}$.
So $\mathcal{I}$ is also model of $Acc(S) \cup \{NAcc(x) | x \in \mathbf{A} \setminus S\} \cup \Sigma_d^0(G) \cup \Sigma_r^0(G)$.

⇐ Let $\mathcal{I}$ be a model of $Acc(S) \cup \{\neg Acc(x) | x \in \mathbf{A} \setminus S\} \cup \Sigma_d^0(G) \cup \Sigma_r^0(G)$. From Proposition 16 on page 40, $S_{\mathcal{I}}$ is a complete extension. As $\mathcal{I}$ satisfies $Acc(S)$, we have $S \subseteq S_{\mathcal{I}}$ and as $\mathcal{I}$ satisfies $\{\neg Acc(x) | x \in \mathbf{A} \setminus S\}$ we have $A \setminus S \subseteq A \setminus S_{\mathcal{I}}$. So $S = S_{\mathcal{I}}$ and S is a complete extension.
Moreover if $\mathcal{I}$ is a model of $Acc(S) \cup \{NAcc(x) | x \in \mathbf{A} \setminus S\} \cup \Sigma_d^0(G) \cup \Sigma_r^0(G)$, as the formulae $NAcc(x) \rightarrow \neg Acc(x)$ belong to $\Sigma^0(G)$, $\mathcal{I}$ is also a model of $Acc(S) \cup \{\neg Acc(x) | x \in \mathbf{A} \setminus S\} \cup \Sigma_d^0(G) \cup \Sigma_r^0(G)$ and we can conclude that S is a complete extension.

□

Proof of Prop. 18 on page 41:

⇒ Assume that S is stable. It is well-known that S is conflict-free. Following the proof of Proposition 5 on page 39, $\exists \mathcal{I}$ model of $\Sigma_0(G)$ with $S_{\mathcal{I}} = S$ and such that:

- $\forall x \in S, \mathcal{I}(Acc(x)) = \textit{true}$ and $\mathcal{I}(NAcc(x)) = \textit{false}$;

- $\forall x \notin S, \mathcal{I}(Acc(x)) =$ *false* and $\mathcal{I}(NAcc(x)) =$ *true*.

We have to prove that $\mathcal{I}$ satisfies the formula $\mathbf{P_{sta}}$: $(\forall c)(\neg Acc(c) \rightarrow (\vee_{b \in \mathbf{R}^-(c)} Acc(b)))$.
Let $c \in \mathbf{A}$. If $c \in S = S_{\mathcal{I}}$, $\mathcal{I}(Acc(c)) =$ *true* and $\mathcal{I}(NAcc(c)) =$ *false* so the implication $\neg Acc(c) \rightarrow (\vee_{b \in \mathbf{R}^-(c)} Acc(b))$ is trivially true.
If $c \notin S$, as S is stable, there exists $a \in S = S_{\mathcal{I}}$ such that $a \in R^-(c)$. As $\mathcal{I}(Acc(a)) =$ *true*, so the implication $\neg Acc(c) \rightarrow (\vee_{b \in \mathbf{R}^-(c)} Acc(b))$ is still true.

$\Leftarrow$ Assume that $\exists \mathcal{I}$ model of $\Sigma_0(G) \cup \{\mathbf{P_{sta}}\}$, such that $S = S_{\mathcal{I}}$. From Proposition 5 on page 39, S is conflict-free.
Let $c \notin S$, as $S = S_{\mathcal{I}}$ $\mathcal{I}(Acc(c)) =$ *false*. As $\mathcal{I}$ satisfies $\mathbf{P_{sta}}$, $\mathcal{I}$ satisfies $(\vee_{b \in \mathbf{R}^-(c)} Acc(b))$. So there exists $b \in \mathbf{R}^-(c)$ such that $\mathcal{I}(Acc(b)) =$ *true*, which means that $b \in S_{\mathcal{I}} = S$. We have proved that S is stable. Then, we conclude that S is stable.

□

Proof of Prop. 19 on page 42: The proof is analogous to the proof of Proposition 17 on page 41. □

Proof of Prop. 20 on page 42:

$\Rightarrow$ Assume that S is stable. It is well-known that S is a naive extension. From Proposition 8 on page 39 $Acc(S)$ is a maximal (for set-inclusion) subset of $Acc(\mathbf{A})$ consistent with $\Sigma^0(G)$. Let $a \notin S$. As S is stable, there exists $x \in S$ such that x attacks a. So $\Sigma^0(G)$ contains the formulas $Acc(x) \rightarrow NAcc(a)$ and $NAcc(a) \rightarrow \neg Acc(a)$. Let $\mathcal{I}$ be a model of $Acc(S) \cup \Sigma^0(G)$, $\mathcal{I}(Acc(x)) =$ *true* (as $x \in S$) so $\mathcal{I}(NAcc(a)) =$ *true*. We have proved that $(\Sigma^0(G) \cup Acc(S)) \vdash NAcc(a)$.

$\Leftarrow$ Assume that $Acc(S)$ is a maximal for set-inclusion subset of $Acc(\mathbf{A})$ consistent with $\Sigma^0(G)$ and $\forall a \notin S, (\Sigma^0(G) \cup Acc(S)) \vdash NAcc(a)$. Let us prove that S is stable.
From Proposition 8 on page 39, S is a naive extension so S is conflict-free.
Let $a \notin S$. We have $(\Sigma^0(G) \cup Acc(S)) \vdash NAcc(a)$. Due to the form of $\Sigma^0(G)$, there exists x such that $(\Sigma^0(G) \cup Acc(S)) \vdash Acc(x)$ and $\Sigma^0(G)$ contains the formula $Acc(x) \rightarrow NAcc(a)$. In that case, we know that x attacks a. Moreover as $Acc(S)$ is a maximal for set-inclusion subset of $Acc(\mathbf{A})$ consistent with $\Sigma^0(G)$, x must belong to S. So we have proved that S is stable.

□

Proof of Prop. 21 on page 42:

$\Rightarrow$ Assume that S is stable. It is well-known that S is conflict-free. From Proposition 5 on page 39, $\exists \mathcal{I}$ model of $\Sigma^0(G)$ such that $S_{\mathcal{I}} = S$.
Let $\mathcal{J}$ be a model of $\Sigma^0(G)$ such that $S \subseteq S_{\mathcal{J}}$. Let $a \notin S$. As S is stable, there exists $x \in S$ such that x attacks a. So $\Sigma^0(G)$ contains the formulas $Acc(x) \rightarrow NAcc(a)$ and $NAcc(a) \rightarrow \neg Acc(a)$. $x \in S \subseteq S_{\mathcal{J}}$ so $\mathcal{J}(Acc(x)) =$ *true*. As $\mathcal{J}$ is a model of $\Sigma^0(G)$, we have $\mathcal{J}(NAcc(a)) =$ *true*.

$\Leftarrow$ Assume that $\exists \mathcal{I}$ model of $\Sigma^0(G)$, such that $S = S_{\mathcal{I}}$. From Proposition 5 on page 39, S is conflict-free.
Let us prove that $\forall a \notin S, (\Sigma^0(G) \cup Acc(S)) \vdash NAcc(a)$. Consider $a \notin S$. Let $\mathcal{J}$ be a model of $\Sigma^0(G) \cup Acc(S)$. Then $S \subseteq S_{\mathcal{J}}$. Due to the assumption, we have $\mathcal{J}(NAcc(a)) =$ *true*. So we have proved that $(\Sigma^0(G) \cup Acc(S)) \vdash NAcc(a)$. Then, due to Proposition 20 on page 42, we conclude that S is stable.

□

Proof of Prop. 22 on page 44:

⇒ Let $\mathcal{I}$ be a model of $\Sigma^0(G) \cup \{\mathbf{P5}\}$. If $\mathcal{I}$ is not $NAcc(\mathbf{A})$-minimal, there exists $\mathcal{I}'$ model of $\Sigma^0(G) \cup \{\mathbf{P5}\}$ such that $N_{\mathcal{I}'} \subset N_{\mathcal{I}}$ and $S_{\mathcal{I}'} = S_{\mathcal{I}}$.
So there exists $x \in N_{\mathcal{I}} \setminus N_{\mathcal{I}'}$. Then we have $\mathcal{I}(NAcc(x)) = \textit{true}$ and $\mathcal{I}'(NAcc(x)) = \textit{false}$.
As $\mathcal{I}$ satisfies **P5**, there exists $y \in \mathbf{R}^-(x)$ such that $\mathcal{I}(Acc(y)) = \textit{true}$. As $S_{\mathcal{I}'} = S_{\mathcal{I}}$, we also have $\mathcal{I}'(Acc(y)) = \textit{true}$. As $\mathcal{I}'$ is a model of $\Sigma^0(G)$, and $y \in \mathbf{R}^-(x)$, we must have that $\mathcal{I}'(NAcc(x)) = \textit{true}$, which is in contradiction with the assumption $x \notin N_{\mathcal{I}'}$.
So $\mathcal{I}$ is a $NAcc(\mathbf{A})$-minimal model of $\Sigma^0(G)$.

⇐ Let $\mathcal{I}$ be a $NAcc(\mathbf{A})$-minimal model of $\Sigma^0(G)$. Let us prove that $\mathcal{I}$ satisfies **P5**. If it is not the case, there exists c such that $\mathcal{I}(NAcc(c)) = \textit{true}$ ($c \in N_{\mathcal{I}}$) and for each $b \in \mathbf{R}^-(c)$, $\mathcal{I}(Acc(b)) = \textit{false}$. As $\mathcal{I}(NAcc(c)) = \textit{true}$, we have $\mathcal{I}(Acc(c)) = \textit{false}$.
Let us consider the interpretation $\mathcal{I}'$ such that $S_{\mathcal{I}'} = S_{\mathcal{I}}$ and $N_{\mathcal{I}'} = N_{\mathcal{I}} \setminus \{c\}$. We have $\mathcal{I}'(NAcc(c)) = \textit{false}$. It is easy to see that I' satisfies the formula $\mathbf{P}^0_{\mathbf{acc}}$. So $\mathcal{I}'$ is a model of $\Sigma^0(G)$. That contradicts the fact that $\mathcal{I}$ is a $NAcc(\mathbf{A})$-minimal model of $\Sigma^0(G)$.

□

B.4 Proofs of Section 5.2 on page 44

All the results obtained in Section 5.1 on page 33 can be extended in a straightforward way, replacing $\Sigma^0(G)$ (resp. $\Sigma^0_d(G)$, $\Sigma^0_r(G)$, $\Sigma^0_s(G)$) by $\Sigma_{as}(G)$ (resp. $\Sigma_d(G)$, $\Sigma_r(G)$, $\Sigma_s(G)$). The proofs can be adapted using c_α and s_α for denoting arguments.
We just illustrate this mechanism by giving the proof for the first item of Proposition 25 on page 48, considering the simplified form of $\Sigma_{as}(G)$ (the proof of propositions 23 on page 46 and 24 on page 47, the proof of the other items of Proposition 25 on page 48 and the proof of Proposition 26 on page 48 can be obtained in a similar way).

Proof of Prop. 25 on page 48:

1. ⇒ Assume that S is conflict-free. Let us consider an interpretation $\mathcal{I}$ of $\Sigma_{as}(G)$ such that : $\mathcal{I}(Acc(x)) = \textit{true}$ if and only if $x \in S$, $\mathcal{I}(NAcc(x)) = \textit{true}$ if and only if $\mathcal{I}(Acc(x)) = \textit{false}$. $\mathcal{I}$ is a model of $Acc(S)$. It remains to prove that $\mathcal{I}$ is a model of $\Sigma_{as}(G)$. Obviously $\mathcal{I}$ satisfies the formulae $(NAcc(c_\alpha) \rightarrow \neg Acc(c_\alpha))$. So, if $\mathcal{I}$ is not a model of $\Sigma_{as}(G)$, there exists a formula $Acc(s_\alpha) \rightarrow NAcc(c_\alpha)$ that is not satisfied by $\mathcal{I}$. In that case $\mathcal{I}(Acc(s_\alpha)) = \textit{true}$ and $\mathcal{I}(NAcc(c_\alpha)) = \textit{false}$. By definition of $\mathcal{I}$, we also have $\mathcal{I}(Acc(c_\alpha)) = \textit{true}$, and then s_α and c_α are in S. So that is in contradiction with S being conflict-free.

   ⇐ Let $\mathcal{I}$ be a model of $Acc(S) \cup \Sigma_{as}(G)$. We prove that $S_{\mathcal{I}}$ is conflict-free. If it is not the case there exist $a, b \in S_{\mathcal{I}}$ and α with the formulae $a = s_\alpha$, $b = c_\alpha$, $Acc(s_\alpha) \rightarrow NAcc(c_\alpha)$ and $(NAcc(c_\alpha) \rightarrow \neg Acc(c_\alpha))$ in $\Sigma_{as}(G)$. We have $\mathcal{I}(Acc(s_\alpha)) = \textit{true}$ and $\mathcal{I}(Acc(c_\alpha)) = \textit{true}$, so there is a contradiction with $\mathcal{I}$ being a model of $\Sigma_{as}(G)$.

□

B.5 Proofs of Section 6.2 on page 54

Proof of Prop. 28 on page 56: By definition, S is a conflict-free extension of AS in the sense of Def. 26 on page 56 means that there is $\Gamma \subseteq \mathbf{R}$ such that (S, Γ) is a conflict-free structure of AS. As no attack of $\mathbf{R}$ is attacked, we have $\Gamma = \mathbf{R}$. Then S conflict-free means that $\forall a, b \in S$, $\nexists \alpha \in \mathbf{R}$ such that $s(\alpha) = a$ and $t(\alpha) = b$, which exactly means that S is a conflict-free subset of $\mathbf{A}$ in Dung's sense. □

Proof of Prop. 29 on page 58:

⇒ Let a be acceptable wrt $U = (S, \Gamma)$. Let $\beta \in \mathbf{R}$ such that $t(\beta) = a$. Either β is inhibited wrt U, or $s(\beta)$ is defeated wrt U. As attacks in $\mathbf{R}$ are simple, we have that $s(\beta)$ is defeated wrt U. So there exists $\gamma \in \Gamma \subseteq \mathbf{R}$ such that $s(\gamma) \in S$ and $t(\gamma) = s(\beta)$. That means that a is acceptable wrt S in Dung's sense.

⇐ Let a be acceptable wrt $S \subseteq \mathbf{A}$ in Dung's sense. Let $\beta \in \mathbf{R}$ such that $t(\beta) = a$. Then $s(\beta)$ is attacked by an argument in S (the corresponding attack being in $\mathbf{R}$). So $s(\beta)$ is defeated by the structure $U = (S, \mathbf{R})$. That proves that a is acceptable wrt the structure $(S, \mathbf{R})$.

□

Proof of Prop. 30 on page 58:

⇒ Let S be an admissible extension of AS in the sense of Def. 29 on page 58. By definition, there exists an admissible structure (S, Γ) of AS. So (S, Γ) is conflict-free. Due to Proposition 28, S is conflict-free in Dung's sense. Each $x \in S$ is acceptable wrt (S, Γ), so it is also acceptable wrt S in Dung's sense, due to Proposition 29 on page 58. It follows that S is an admissible subset of $\mathbf{A}$ according to Dung's definition of admissibility.

⇐ Let S be an admissible subset of $\mathbf{A}$ in Dung's sense. Due to Proposition 28, there exists $\Gamma \in \mathbf{R}$ such that (S, Γ) is a conflict-free structure. As attacks in $\mathbf{R}$ are simple, conflict-freeness implies that $\Gamma = \mathbf{R}$.
Moreover, each $x \in S$ is acceptable wrt S in Dung's sense, so, due to Proposition 29, each $x \in S$ is acceptable wrt $(S, \mathbf{R})$. As attacks in $\mathbf{R}$ are not attacked, it follows that the structure $(S, \mathbf{R})$ is admissible. So, by definition, S is an admissible extension of AS.

□

Proof of Prop. 31 on page 59:

⇒ Let S be a complete extension of AS in the sense of Def. 31 on page 59. By definition, there exists a complete structure (S, Γ) of AS. As attacks in $\mathbf{R}$ are simple, conflict-freeness implies that $\Gamma = \mathbf{R}$. So, $(S, \mathbf{R})$ is a complete structure, meaning that (i) $(S, \mathbf{R})$ is admissible and (ii) each $x \in \mathbf{A}$ (resp. $x \in \mathbf{R}$) which is acceptable wrt $(S, \mathbf{R})$ belongs to S (resp. $\mathbf{R}$). Due to the above propositions, (i) implies that S is admissible in Dung's sense and (ii) implies that each $x \in A$ which is acceptable wrt S in Dung's sense belongs to S (as it is also acceptable wrt the structure $(S, \mathbf{R})$). So S is a complete extension in Dung's sense.

⇐ Let S be a complete extension in Dung's sense. Due to Proposition 30, the structure $(S, \mathbf{R})$ is admissible.
Let $x \in \mathbf{A}$ such that x is acceptable wrt $(S, \mathbf{R})$, then x is acceptable wrt S in Dung's sense. As S is complete, $x \in S$.
As attacks in $\mathbf{R}$ are not attacked, it follows that the structure $(S, \mathbf{R})$ is complete.

□

Proof of Prop. 32 on page 60:

⇒ Let S be a stable extension of AS in the sense of Def. 33 on page 59. By definition, there exists a stable structure (S, Γ) of AS. So (S, Γ) is conflict-free. As attacks in $\mathbf{R}$ are simple, conflict-freeness implies that $\Gamma = \mathbf{R}$. So, $(S, \mathbf{R})$ is a stable structure.
Due to Proposition 28, S is conflict-free in Dung's sense. It remains to prove that $\forall a \notin S$ a is attacked by S.
Let $a \notin S$, as the structure $(S, \mathbf{R})$ is stable, a is defeated wrt $(S, \mathbf{R})$ which exactly means that a is attacked by S. So S is stable in Dung's sense.

⇐ Let S be a stable extension according to Dung's definition. S is conflict-free in Dung's sense, so the structure $(S, \mathbf{R})$ is conflict-free. Moreover, $\forall a \notin S$ a is attacked by S. As said in the first part of this proof, that exactly means that $\forall a \notin S$ a is defeated wrt $(S, \mathbf{R})$. As the structure $(S, \mathbf{R})$ contains all the attacks, we have proved that $(S, \mathbf{R})$ is a stable structure, so S is a stable extension of AS in the sense of Def. 33 on page 59.

□

Proof of Prop. 33 on page 60: Let $U = (S, \Gamma)$ be a stable structure of ASAF.

- We first prove that U is admissible. By definition, U is conflict-free.
  Let $x \in (S \cup \Gamma)$ such that there is $\beta \in \mathbf{R}$ with $t(\beta) = x$. As U is conflict-free, either $\beta \notin \Gamma$, or $s(\beta) \notin S$. In the first case, β is inhibited wrt U, and in the second case $s(\beta)$ is defeated wrt U. So, we have proved that x is acceptable wrt U. Consequently, U is admissible.
- It remains to prove that each $x \in \mathbf{A}$ (resp. $x \in \mathbf{R}$) which is acceptable wrt (S, Γ) belongs to S (resp. Γ). Let $x \in \mathbf{A}$ being acceptable wrt (S, Γ). Assume that $x \notin S$. As U is stable, x is defeated wrt U. So there is $\beta \in \Gamma$ with $s(\beta) \in S$ and $t(\beta) = x$. As x is acceptable wrt U, either β is inhibited wrt U, or $s(\beta)$ is defeated wrt U. In other words, there is $\gamma \in \Gamma$ such that $s(\gamma) \in S$ and $t(\gamma) \in \{\beta, s(\beta)\}$. That is in contradiction with U being a conflict-free structure. So we have proved that $x \in S$.
  An analogous reasoning holds for proving that each $x \in \mathbf{R}$ which is acceptable wrt (S, Γ) belongs to Γ.

□

B.6 Proofs of Section 6.3 on page 60

Proof of Prop. 34 on page 60: The proof is inspired by the proof of Proposition 5.
Let us recall that $\Sigma(G)$ includes formulae **(3, 4, 5)**.

⇒ Assume that the structure (S, Γ) is weakly conflict-free. Let us define an interpretation $\mathcal{I}$ of $\Sigma(G)$ as follows :
- $\mathcal{I}(Acc(x)) =$ *true* if and only if $x \in S$ and $\mathcal{I}(NAcc(x)) =$ *true* if and only if $\mathcal{I}(Acc(x)) =$ *false*.

- $\mathcal{I}(Val(x)) = \textit{true}$ if and only if $x \in \Gamma$.
We have $S_{\mathcal{I}} = S$ and $\Gamma_{\mathcal{I}} = \Gamma$. It remains to prove that $\mathcal{I}$ is a model of $\Sigma(G)$.
Obviously $\mathcal{I}$ satisfies Formula **(5)**.
If $\mathcal{I}$ does not satisfy Formula **(4)**, there exist $x \in \mathbf{A}$ and $\alpha \in \mathbf{R}$ such that $t(\alpha) = x$, $\mathcal{I}(Val(\alpha)) = \textit{true}$, $\mathcal{I}(Acc(s(\alpha))) = \textit{true}$ and $\mathcal{I}(NAcc(x)) = \textit{false}$.
In other words, $\alpha \in \Gamma$, $s(\alpha) \in S$ and $x \in S$. That is in contradiction with (S, Γ) being weakly conflict-free.
If $\mathcal{I}$ does not satisfy Formula **(3)**, there exist $\alpha, \beta \in \mathbf{R}$ such that $t(\alpha) = \beta$, $\mathcal{I}(Val(\alpha)) = \textit{true}$, $\mathcal{I}(Acc(s(\alpha))) = \textit{true}$ and $\mathcal{I}(Val(\beta)) = \textit{true}$.
In other words, $\alpha, \beta \in \Gamma$ and $s(\alpha) \in S$. That is in contradiction with (S, Γ) being weakly conflict-free.
It follows easily that $\mathcal{I}$ is a model of $\Sigma(G)$.

$\Leftarrow$ Let $\mathcal{I}$ be a model of $\Sigma(G)$. We prove that the structure $(S_{\mathcal{I}}, \Gamma_{\mathcal{I}})$ is weakly 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 **(4)** is falsified. In the second case, Formula **(3)** is falsified. That is in contradiction with $\mathcal{I}$ being a model of $\Sigma(G)$.

□

Proof of Prop. 35 on page 60: The proof is inspired by the proof of Proposition 9.
Let us recall that $\Sigma_d(G)$ is obtained from $\Sigma(G)$ by adding formulae $\mathbf{P}^{\mathbf{arg}}_{\mathbf{def}}$ and $\mathbf{P}^{\mathbf{att}}_{\mathbf{def}}$.

$\Rightarrow$ Assume that the structure (S, Γ) is weakly admissible. Due to the proof of Proposition 34 on page 60, $\exists \mathcal{I}$ model of $\Sigma(G)$ with $S_{\mathcal{I}} = S$ and $\Gamma_{\mathcal{I}} = \Gamma$ and such that:

- $\forall x \in S, \mathcal{I}(Acc(x)) = \textit{true}$ and $\mathcal{I}(NAcc(x)) = \textit{false}$;
- $\forall x \notin S, \mathcal{I}(Acc(x)) = \textit{false}$ and $\mathcal{I}(NAcc(x)) = \textit{true}$.
- $\mathcal{I}(Val(x)) = \textit{true}\ \forall x \in \Gamma$

It is easy to prove that $\mathcal{I}$ satisfies formulae $\mathbf{P}^{\mathbf{arg}}_{\mathbf{def}}$ and $\mathbf{P}^{\mathbf{att}}_{\mathbf{def}}$.

$\Leftarrow$ Let $\mathcal{I}$ be a model of $\Sigma_d(G)$. It is easy to prove that the structure $(S_{\mathcal{I}}, \Gamma_{\mathcal{I}})$ is weakly admissible.

□

Proof of Prop. 36 on page 61: Let us recall that $\Sigma_r(G)$ is obtained from $\Sigma(G)$ by adding formulae $\mathbf{P}^{\mathbf{arg}}_{\mathbf{reins}}$ and $\mathbf{P}^{\mathbf{att}}_{\mathbf{reins}}$.

$\Rightarrow$ Assume that the structure (S, Γ) is complete. Due to the proof of Proposition 35 on page 60, $\exists \mathcal{I}$ model of $\Sigma_d(G)$ with $S_{\mathcal{I}} = S$ and $\Gamma_{\mathcal{I}} = \Gamma$ and such that:

- $\forall x \in S, \mathcal{I}(Acc(x)) = \textit{true}$ and $\mathcal{I}(NAcc(x)) = \textit{false}$;
- $\forall x \notin S, \mathcal{I}(Acc(x)) = \textit{false}$ and $\mathcal{I}(NAcc(x)) = \textit{true}$.
- $\mathcal{I}(Val(x)) = \textit{true}\ \forall x \in \Gamma$

It is easy to prove that $\mathcal{I}$ satisfies formulae $\mathbf{P}^{\mathbf{arg}}_{\mathbf{reins}}$ and $\mathbf{P}^{\mathbf{att}}_{\mathbf{reins}}$.

$\Leftarrow$ Let $\mathcal{I}$ be a model of $\Sigma_d(G) \cup \Sigma_r(G)$. Due to Proposition 35 the structure $(S_\mathcal{I}, \Gamma_\mathcal{I})$ is weakly admissible.
It is easy to prove that each $x \in \mathbf{A}$ (resp. $x \in \mathbf{R}$) which acceptable wrt $(S_\mathcal{I}, \Gamma_\mathcal{I})$ belongs to $S_\mathcal{I}$ (resp. $\Gamma_\mathcal{I}$). As a consequence, $\{\alpha \in \mathbf{R} | \nexists \beta \in \mathbf{R}$ such that $t(\beta) = \alpha\} \subseteq \Gamma_\mathcal{I}$. So the structure $(S_\mathcal{I}, \Gamma_\mathcal{I})$ is admissible and complete.

□

Proof of Prop. 37 on page 61: Let us recall that $\Sigma_s(G)$ is obtained from $\Sigma(G)$ by adding formulae $\mathbf{P}^{\mathbf{arg}}_{\mathbf{sta}}$ and $\mathbf{P}^{\mathbf{att}}_{\mathbf{sta}}$.
Let us first note that if $\mathcal{I}$ is a model of $\Sigma(G)$ which also satisfies Formula $\mathbf{P}^{\mathbf{att}}_{\mathbf{sta}}$, then the structure $(S_\mathcal{I}, \Gamma_\mathcal{I})$ is conflict-free.

$\Rightarrow$ Assume that the structure (S, Γ) is stable. We know that $\exists \mathcal{I}$ model of $\Sigma(G)$ 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*;
- $\forall x \notin S, \mathcal{I}(Acc(x)) =$ *false* and $\mathcal{I}(NAcc(x)) =$ *true*.
- $\mathcal{I}(Val(x)) =$ *true* $\forall x \in \Gamma$

It is easy to prove that $\mathcal{I}$ satisfies formulae $\mathbf{P}^{\mathbf{arg}}_{\mathbf{sta}}$ and $\mathbf{P}^{\mathbf{att}}_{\mathbf{sta}}$.

$\Leftarrow$ Let $\mathcal{I}$ be a model of $\Sigma_s(G)$. From the remark above, the structure $(S_\mathcal{I}, \Gamma_\mathcal{I})$ is conflict-free. Then the fact that $\mathcal{I}$ satisfies $\mathbf{P}^{\mathbf{arg}}_{\mathbf{sta}}$ and $\mathbf{P}^{\mathbf{att}}_{\mathbf{sta}}$ enables to prove that the structure $(S_\mathcal{I}, \Gamma_\mathcal{I})$ satisfies the two following conditions : $\forall a \notin S_\mathcal{I}$, a is defeated wrt $(S_\mathcal{I}, \Gamma_\mathcal{I})$, and $\forall \alpha \notin \Gamma_\mathcal{I}$, α is inhibited wrt $(S_\mathcal{I}, \Gamma_\mathcal{I})$. So the structure $(S_\mathcal{I}, \Gamma_\mathcal{I})$ is stable.

□